Using Melodic Motifs for Automatic Transcription of White Mensural Notation via N-Grams

Tomer Iwan
t.iwan@student.vu.nl
VU Amsterdam
Amsterdam, The Netherlands

Albert Meroño-Peñuela
albert.merono@vu.nl
VU Amsterdam
Amsterdam, The Netherlands

Stefan Schlobach
k.s.schlobach@vu.nl
VU Amsterdam
Amsterdam, The Netherlands

ABSTRACT

Manually transcribing music sheets is a time consuming task, and it is therefore desirable to have a system that performs the task automatically. Optical music recognition (OMR) is a field of research that investigates how to build such a system. The field has seen many developments over the years, but the problem remains far from solved. This research focuses on building a system capable of automatically translating music scores written in early notation. While traditional methods perform symbol detection once, this paper performs it twice. The first detection is carried out in order to recognise the musical symbols, and the second to detect the note heads. Doing so should achieve a greater performance in recognising the notes. A novel approach is introduced by supplying the process with knowledge about motifs derived from N-grams. These N-grams form the patterns in pieces and are used to train a Random Forest (RF) model. The output of the OMR system is then parsed into the model in order to catch errors. The output of the OMR system is then fed into the model in order to fill in missing notes. Furthermore, a second approach is introduced, by means of n-gram matching using a sliding window approach. In this paper, both N-grams derived from a gold standard and from the output from the OMR system are researched. While both approaches in N-gram processing improve the system, a clear difference in performance is not found, both using gold standard N-grams and those derived from the OMR output. However, the N-grams derived differ from each other heavily. This paper shows promises in not only the usage of pattern recognition, but also in fully automating the process of inserting knowledge into OMR systems.

CCS CONCEPTS

• **Information systems → Music retrieval**; • **Applied computing → Digital libraries and archives**; *Document capture.*

KEYWORDS

Optical Music Recognition, N-grams, Pattern Recognition, Random Forest, Sliding Window, White Mensural Notation

ACM Reference Format:
Tomer Iwan, Albert Meroño-Peñuela, and Stefan Schlobach. 2019. Using Melodic Motifs for Automatic Transcription of White Mensural Notation via N-Grams. In *Proceedings of DFLM 2019 (DFLM '19).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Music analysis is a large part of the interdisciplinary field of Music Information Retrieval (MIR) [6]. As its name implies, MIR aims to retrieve information from music. Researchers in the field of MIR have a great interest in the diversification of the music available, for expanding their knowledge in music information. This is were the technique of Optical Music Recognition (OMR) proves to be essential. It is conceptually similar to Optical Character Recognition (OCR). While OCR focuses on the retrieval of text, OMR looks at music notation. The main benefit of applying OMR is that its aim is to automatically process scores and give a machine readable representations as output.

The work presented in this paper focuses on music scores written in White Mensural Notation (WMN), the standard notation for choral music in the Renaissance period. Mensural notation introduced the duration of notes, in numerical proportions, based on different shapes. Nowadays, this notation is no longer used to write music. However, the existing scores are of such interest that it is useful to transcribe them, either for performance or study, using Common Music Notation (CMN). Digitization of these prints proves very valuable for the online preservation of music, making them available for a wide range of applications within music analysis. In this paper, our musical piece of interest is an important prayer from the Bible, *Kyrie Eleison*, due to its repeating nature. A characteristic of this piece is that for the first number of iterations, the melody is repeated up until the last line. On this last line, a variation is used. The choice to use a variation near the end is for a smoother transition into the following prayer, *Gloria*.

The aim of this paper is to develop an OMR framework that leverages knowledge in motif repetition, based on machine learning models and generating N-grams for building a predictive model. The aim of the predictive N-gram model is to improve the machine learning model in its task. Therefore the following two high level research questions will be addressed:

- How does an object detection model perform in the task of OMR in White Mensural Notation?
- What impact do N-grams that model motif repetition have on the performance of the transcription?

The second research question can be divided into four sub-questions, namely:

- What impact do the N-grams, derived from the gold standard, have on the performance of the model, using random forest algorithm?
- What impact do the predicted N-grams, from the output of the OMR tool, have on the performance of the model, using random forest algorithm?
- What impact do the N-grams, derived from the gold standard, have on the performance of the model, using a sliding window approach?
- What impact do the N-grams, derived from the output of the OMR tool, have on the performance of the model, using a sliding window approach?

This paper is organized in six sections. The paper begins with a literature overview, showcasing the related work in the field of OMR and pattern recognition in section 2. The properties of White Mensural Notation are laid out in section 3. The next section (Section 4), goes into detail on the general framework that earlier work mainly followed. Then in section 5, the methods are discussed, including the proposed framework of the software. Next the experimental setup is presented in section 6. The results are presented in section 7, and we discuss them in Section 8. Furthermore, the discussion will suggest proposals for future work to improve the system. Finally, the paper concludes by showcasing the key findings, in Section 9.

2 RELATED WORK

In this section, an overview is provided for the work done in the fields of OMR and pattern recognition.

2.1 OMR

The vast majority of research has been done on music written in modern notations. Despite the differences between modern and early prints, the work done on the former is of great importance for the understanding the current developments in the field.

One such a tool, for the recognition of modern music notation, is Gamut. Gamut, built on the Gamera framework [7], is based on the use of K-Nearest Neighbours (KNN). Gamut relies on the removal of staff lines in order to retrieve the pitch of the detected notes. This step in preprocessing is performed by the MusicStaves toolkit, which is also based on the Gamera framework.

An OMR tool specifically created to detect objects in early typographic print, is Aruspix [17]. Their method is based on the use of Hidden Markov Models on the staff level. Aruspix foregoes the use of staff-line removal, as it considers this phase to be difficult and unreliable [18]. Another recognition tool for scores written in White Mensural Notation was developed by [21]. Like Gamut, this tool relies on the use of KNN. However, this tool assumes an RGB relation to hold for non-musical symbols, as one of its stages includes the removal of the frontispieces, which in their case was colored.

Finally, MuRET [20] is an OMR tool designed to transcribe handwritten scores in early notation. It holds the assumption that no approach will lead to perfect transcription, and therefore human intervention plays an integral role in this system.

2.2 Pattern Recognition

A lot of work has been done on retrieving musical patterns. Patterns are defined as sequences of notes that occur together frequently in a piece. For this application, the use of N-grams has proven to be a reliable method of identifying the musical patterns [16]. The preferred method of constructing said N-grams is by means of the sliding window algorithm. These sequences can be fully dependent, semi-dependent or fully independent of the context. In the first situation, the N-grams could be different depending on the key signatures, for instance. One step closer to independence is to take only into account the clef in which the pattern was detected. Finally, a fully independent approach is to only look at the patterns in terms of the relative position of the notes on the lines with respect to the last note, represented as positive/negative integer half-steps; instead of an absolute pitch representing each note. This method yields true pitch invariant patterns, able to be extrapolated to various pieces, independent of their key signature or clef (see examples in Tables 2 and 3).

2.3 Data Sets

Not many data sets are available for early prints. The SEILS data set [15] is one of the few to create such a data set. Their data set contains Italian madrigals from the 16th century, which are available in a variety of representations. Aside from the original prints, the project manually created modern representations of these pieces, written in Lilypond, MIDI, and Music XML.

3 WHITE MENSURAL NOTATION

Music can be represented in various ways. While there exist many similarities between WMN and CMN, there are some crucial differences. The aim of this section is to display the properties and present the modern representation when appropriate. An example of WMN is shown in Figure 2.

The printed form of a musical work is called a **score** or sheet music which consists of notes. Scores are made of staves, which consist of the five lines and the spaces around them. The **clefs** of interest, in this paper, are the alto, bass, tenor, and treble clefs.

The note length or **duration** is decided by the representation of said note. The **semi-breve** corresponds to the modern whole note. A **breve** is twice the length of the semi-breve. The **minima**, **semi-minima**, **fusa**, and **semi-fusa**, respectively hold the note values of a half, quarter, eighth and sixteenth of a whole note. **Dots** extend the duration by half the original length.

Frontispieces are decorative illustrations at the beginning of pieces.

The **key signature** designates that notes are to be played either higher or lower than the corresponding notes. Most commonly it is a set of either sharps or flats placed together on the staff, at the beginning of a line, following the clef. They can appear also later on, notable after a double bar line. The key signature applies up until the end of the piece or up to the next key signature. The key is either in the major or minor mode. The **custos** appears at the end of the stave, and indicates on which line the following note on the next stave will occur. No equivalent exists in modern notation.

The sign of **congruence**, or cadence point, indicates whether a canon starts or ends.

The **corona** (also known as Fermata) is a symbol that indicates that a note is to be played longer beyond what is written. Unclear is how much longer it should be played, and therefore is up to the performer to decide. The most common interpretation is to play a note twice as long. Time signatures appear at the beginning, following the clef and key signature, and specify the number of beats contained in each bar. Rests are intervals of silence in music, marked by symbols indicating their length. Each rest symbol corresponds with a particular note value.

Finally the **accidentals** include the **sharp**, **flat** and **natural symbols**. Accidentals are always precede the notes they affect. The sharp raises the pitch by a semitone, while the flat lowers the pitch by a semitone. The natural symbol undoes the effect of the other accidentals as it reverts the notes back to the natural note.

4 GENERAL FRAMEWORK

Based on the the related work presented above, a general framework is identified by [19], consisting of the following four phases.

(1) Preprocessing
(2) Segmentation
(3) Object Recognition
(4) Semantic Reconstruction

4.1 Preprocessing

First the input images need to be processed in order for them to be usable in the following phases. Many methods exist for this stage. Skew correction can be applied in order to straighten images. Background noise removal is also an important technique in order to remain only with useful information. Some documents may be unreadable, due to degraded conditions. This applies in particular to early prints. These images can be subjected to contrast enhancement techniques.

4.2 Segmentation

This stage aims to extract the symbols and their position from the processed music sheet. This process starts with the extraction of sub-images from the main image. These images represent the music staves. Next, these images are subjected to the object segmentation process. A divide exists in the importance of staff-line removal. Some articles consider them an important step in the segmentation process [3, 17], while others prefer to remove them in favour of noiseless object extraction [7]. In order to identify these lines, retrieving horizontal projections is the method of choice in the majority of tools. [4].

4.3 Object Recognition

Regardless of the method of segmentation chosen, object recognition is applied on the symbols. To recognize which symbol is displayed, a model must be trained on instances. The image of the symbols is parsed as the input, and the model gives an output containing the symbol name. The current trend is to use deep learning for this task. [14]

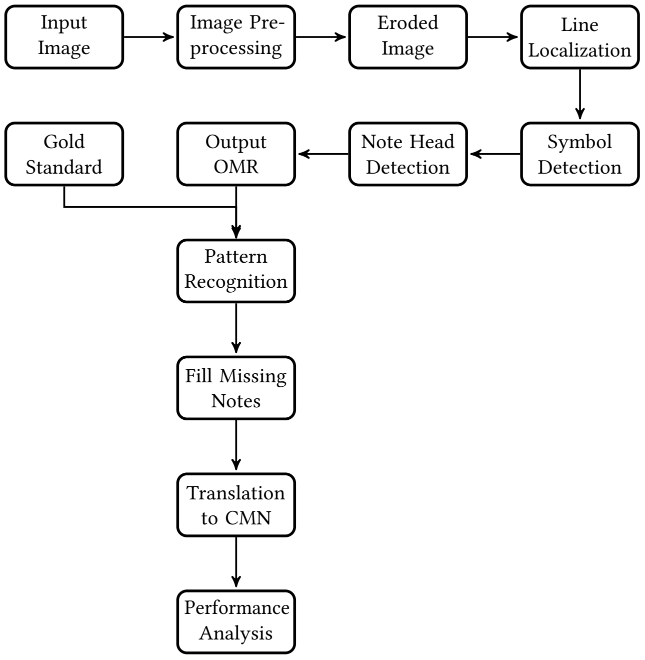

Figure 1: The full pipeline.

4.4 Semantic Reconstruction

Based on the detected objects in the previous stage, the next step is to derive the relationships between these objects. The relationships can be either logical or implied. Logical relationships are for example the notion that a note head and a stem together form the note. The implied relationships include the combination of notes with accidentals.

5 METHODOLOGY

In this section, the methodology is described. Before delving into the individual parts of the constructed pipeline, a global overview is provided.

(1) Image Preprocessing
(2) Staff-line Localization
(3) Symbol Detection
(4) Note Head Detection
(5) Data Analysis
(6) N-gram Predictions
(7) Encoding

As can be seen, the pipeline deviates slightly from the one presented in section 4. The main difference is in the implementation of knowledge by means of N-grams.

Also, splitting the object detection stage into symbol and note head detection deviates from the general framework. The full pipeline is provided in Figure 1.

5.1 Building the Data Set

First of all, a data set needed to be built, as a suitable set was not found. This set was built manually using three music scores as data sources. These scores were from the Kyrie Eleison piece, each one written in a different clef. One score was written in the Alto clef, another score in Bass clef, and the last one in Treble. Each symbol was labelled individually. Annotating was done by using [2] as a guideline. Furthermore, a second data set was created by only taking the heads of the notes into account. All other objects were therefore omitted. Finally, both data sets were divided into train and test splits. before training.

5.2 Image Preprocessing

The original images of the music scores had to be processed in order for them to be fit for use during the later stages of the pipeline. Figure 2 shows an example of an image which is used as input in the system.

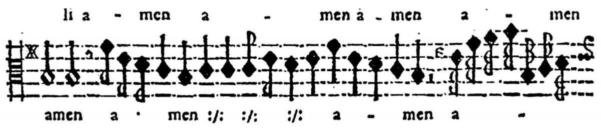
li a - men a - men à - men a - men
amen a - men :/: :/: :/: a - men a -

Figure 2: The input image.

As many images had low contrast, it was hard to distinguish open note heads from closed ones. Furthermore, the majority of staff lines were either crooked or were interrupted by white spaces, which made line detection difficult. The solution for these problems were largely solved by applying the following steps.

During the preprocessing stage, images were turned into grayscale images. Using Otsu's method [13], these images were then reduced to binary images. Lastly, the properties derived from this method were used to erode the original images. See Figure 3.

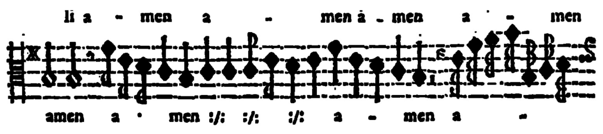
li a - men a - men à - men a - men
amen a - men :/: :/: :/: a - men a -

Figure 3: The eroded input image.

5.3 Staff-line Localization

After the image preprocessing stage was done, the lines needed to be detected. In order to retrieve these lines, the Hough transform [8] technique was applied. See Figure 4.

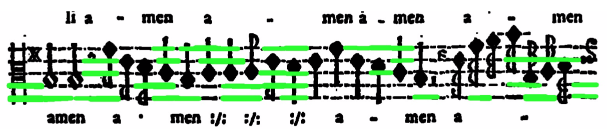
li a - men a - men à - men a - men
amen a - men :/: :/: :/: a - men a -

Figure 4: Detected Hough lines.

This stage was considered extremely critical, as ultimately, the line height was the decisive factor for determining the notes.

Issues occurred using this method, as the image quality of the scores fluctuated heavily. This caused the algorithm to find the wrong number of lines. Therefore, rules were created in order to retrieve exactly five lines per stave.

These rules were based on the gaps between the identified lines. This proved necessary, as the Hough transformation could not differentiate between detected horizontal lines on the same height.

These lines were therefore excluded based on the minimal line gap requirement. This method resulted in a great improvement in line localization in images.

The only issue remaining, was that the transformation could not recognize all the lines in an image, resulting in less than five lines. The solution which reduced the occurrence of this issue, was to erode the image again, until five lines were detected.

The next step was to calculate the differences between the five identified lines. The average value was determined of the difference, effectively providing a true line gap. This value was then divided by two, in order to visualize the implied lines. These lines include those that are between the lines and those outside of the staves. The latter lines are also known as ledger lines. This step was crucial for determining the values of notes whose heads were not located on top of one of the five lines. Finally, all the lines were visualized on the original image. See Figure 5.

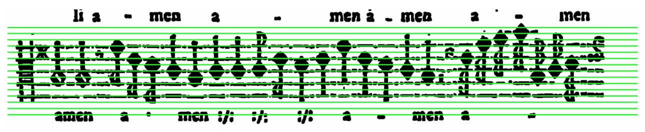
li a - men a - men à - men a - men
amen a - men :/: :/: :/: a - men a -

Figure 5: The derived lines.

5.4 Symbol Detection

After preprocessing the images and retrieving the position of the lines, the first object detection was ready to be employed. This detection was performed by training the Faster-R-CNN model from the TensorFlow [1] framework on all the symbols that were present in the original images. Training lasted for approximately 10.000 epochs. The output of this model contained the class names of the detected symbols and the coordinates of their bounding boxes. See Figure 6.

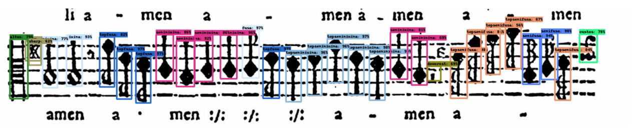
li a - men a - men à - men a - men
amen a - men :/: :/: :/: a - men a -

Figure 6: Bounding boxes surround the detected symbols.

5.5 Note Head Detection

A second object detection model was trained in order to retrieve the position of the note heads. For this task, an extended method of Faster R-CNN was used, namely Mask R-CNN [10]. This method enabled the use of segmentation masks. This time, training lasted only five epochs, mainly due to the simplicity of the task. See Figure 7.

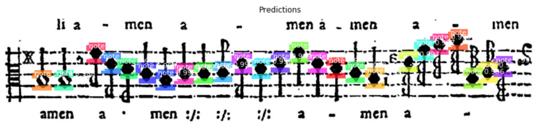
Predictions
li a - men a - men a - men a - men
amen a - men :/: :/: :/: a - men a -

Figure 7: Bounding boxes surround the detected note heads.

5.6 Data Analysis

This subsection contains the bulk of the work done on the project. Described here is how a small amount of data was transformed in meaningful features used to understand what was detected in the previous two stages. Some of these steps require basic knowledge of music. Therefore, an overview of the properties is provided in appendix 3.

After detecting both the symbols and the note heads, it was possible to start working with the collected data. The data was processed using the Pandas [12] package in Python.

5.6.1 The Individual Data Frames. Two data frames were created based on the two object detection stages. The symbol detection offered the following essential pieces of information; the coordinates of the bounding boxes and the names of the classified items. Using this basis, it was possible to derive new features.

In order to represent the data in chronological order, it was first necessary to retrieve the x-values of the detected symbols. This was possible by calculating the average coordinate values over the lower left and lower right corner of the bounding boxes. These values were then stores under the column named 'x'. Now it was possible to sort the data frame on the value of the x coordinates.

After sorting, it was necessary to remove objects from the data frame which were detected twice. This was done in order to prevent the occurrence of overlapping bounding bounding boxes on the same symbol. A calculation was done on the values of x, in order to retrieve the average distance between symbols. Next, a grouper function was applied that looked whether the distance between two objects was equal to or larger than the average. If the objects were decided to be in too close of a vicinity of each other, and the class name was similar, then they were deemed to be the same object, detected twice. The coordinates of the bounding boxes were then averaged by the grouper function, effectively removing the duplicate observation.

The next step concerned the acquisition of the y-values. This was done by calculating the average over the values of the upper right and lower left corner of the bounding boxes.

The following step was to create new columns and occupy them with the surrounding objects of the current one. These columns were created by shifting the column containing the current symbol.

This was an important step for identifying which notes were to be affected by accidentals and dots. Again, additional columns were added to the data frame. This time, they contained information whether a note was preceded by either a sharp or a flat, or that it was followed by a dot.

Based on the note names, the note length values were derived. A semi-breve (a whole note) was assigned the note value 1. Accordingly, the rest of the note classes were assigned values relative to the value of the semi-breve. Therefore the minima, semi-minima, fusa, and semi-fusa respectively got assigned the values 1/2, 1/4, 1/8, and 1/16. Finally, the breve got assigned the value 2, as having the value of twice a whole note. With this step, the first data frame was completed.

As mentioned earlier, the second data frame originated from the note head detection results. Again, the data frame contains information on the coordinates of the bounding boxes. In the same fashion, the x and y values were derived, and the data frame was sorted on the former values.

5.6.2 The Merged Data Frame. Next, the two data frames were joined on the x-values. If the x-values were in close vicinity to each other then it was decided to be the same object, i.e. a note. The decision was based on whether the distance between the objects was smaller than the computed average distance.

In the case of notes, the average y-value was taken from the note head data frame. Now it was necessary to compare the heights of the notes with the staff lines computed earlier. Based on the detected clef, rules were set to determine which note was displayed. Using a for loop, each line, in the list of lines, was compared to the value of y. The line having the lowest absolute difference to the y-value, determined on which line the note head was placed. This was, logically, only performed for the notes, as other symbols were independent of their height relative to the lines.

The line heights of the notes were then added to the data frame and used to determine the note. A function was built that took into consideration both the line height of the note and the detected clef of the piece. Based on that, rules were written to retrieve the notes.

The final step, was to represent the key elements. This was also done by applying a function. As explained in Appendix 3, the key signature contains either a set of sharps or flats at the beginning of the line. Therefore, it was necessary to differentiate between accidentals used on individual notes and those used for the key signature. Based on the x values, it was determined whether the accidentals were used as key elements or not. Also, given the index of the first note of the line in the data frame, it was possible to exclude the sharps and flats that were not part of the key signature. The effect of the key signature to move natural notes either one semitone higher or lower was included. Also a note written to be a semitone lower became a natural note, and vice versa, if it was part of the notes affected by the key signature.

The effect of dots was also taken into account using a function that multiplied the length of the note by one and a half each time a note was followed by a dot.

5.7 N-gram Predictions

During the last phase of the pipeline, the N-gram algorithm was applied. The N-grams were derived from a fourth music score, Kyrie Eleison written in the Tenor clef. Two N-gram vocabularies were built, one from the generated pieces, while the other originated from the gold standard.

N-grams consisted of pitch-invariant notes, in order to focus solely on the patterns of the note placements. These n-grams were then used to train the random forest algorithm. Based on the surrounding symbols, occupying the columns of the data frame, the algorithm gave an output in which it aimed to predict the current symbol.

The sequence of notes was determined by means of keys of a piano. The distance between keys was determined, which made up the sequential patterns of interest. For instance a C got assigned the value of zero, while a C-sharp got assigned the value one, giving a difference of one between them.

After retrieving the relative distance between notes, it was necessary to retrieve N-grams. Chosen was to take 4-grams. This was done by shifting the column in the data frame into both directions of preceding and following notes.

The same process was used to retrieve the gold patterns, as the gold standard objects were also added to a new data frame. The choice for the random forest algorithm was made because it is an ensemble algorithm. Multiple decision trees processed the data and the end result was therefore the object with the highest weighted prediction. Not taken into account in the patterns, were the rests, dots, note durations and ties. A second method was employed to retrieve suggestions for insertions. A sliding window approach was used to match the patterns to the output. Using a set Levenshtein of distance one, the strings were compared. This was done both using the N-grams derived from the gold standard, as from those derived from the output of the OMR tool.

The output of the OMR tool was analysed and the empty slots were extracted to build another data frame. Then 4-grams were derived, with the missing note in each of the available slots. To denote the missing note, the number 1000 was filled in the empty slot. Then these 4-grams are compared to the patterns found in either the gold standard string or the output of OMR itself to find patterns with a Levenshtein distance of exactly one. Namely, the anomaly in the pattern being the 1000 spot. When a match was found, it was replaced in the original query, only in those spots where a note was missing. This to ensure that wrong insertion would not occur. For the sliding window approach, the ten most occurring patterns were taken into account.

5.8 Encoding

The final step was to transform the sequence into tinynotation. Tinynotation is a module of the Music21 package [5] which allows users to create music representations from strings. In order to conform to the convention of this module, the pandas series first needed to be rewritten. This was done by defining a function that transformed the notes to their representation in tinynotation. This included appending the note length, accidentals, dots, and ties directly to the note names.

Then the series needed to be converted to a string, and be appended to a second string containing: 'tinynotation: '. Now it was possible to parse the string and obtain both a visual and auditory representation of the generated music piece. The final visual representation is displayed in Figure 8.

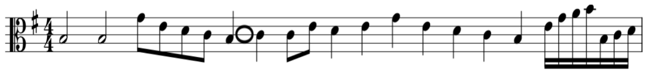

Figure 8: The translated sequence in Music21. Note that the one missing note is highlighted with a black circle.

Applying the knowledge derived from the pattern recognition phase, the empty slots were filled. Every note that was missing was extracted from the main data frame and they were added to a new data frame. In this data frame the missing note was determined using the pattern, using either the previous note or the next note. The predicted distance was taken into consideration relative the note of interest and using a list containing all possible notes, the missing note was to be filled in.

For example, the note F2-sharp holds index number 7, in the list of notes. If a distance was predicted of minus 2, the index was deducted by that number, therefore index 9, G2-sharp was to be filled in. For instances that did not have a note preceding them, the prediction relied on the next note instead. In such cases, the predicted distance was added to the index of the next note, instead of deducted.

6 EXPERIMENTS

In order to gauge the performances of the methods, the same set of pages were evaluated. This was mainly put on the detection and recognition of the individual symbols. The gold standard for these pages was provided by the thesis supervisor. The following three experiments were of interest.

First, the piece generated by the OMR tool was compared to the gold standard. The second experiment concerned the performance of the OMR tool combined with the pattern recognition machine learning model. Both the gold standard and the own output were used to train separate models. Again, the final output was compared to the gold standard.

The last experiment used the pattern recognition model based on the sliding window approach. Again the output was compared to the gold standard, and the patterns were derived from either the gold standard or the OMR output.

The full repository for this work can be found on GitHub[1].

7 RESULTS

In order to evaluate the predicted sequence with the gold standard, we opted to use the Levenshtein distance metric [11]. The algorithm compares to strings in order to determine the minimal amount of edits form one to the other. Edits in this context include insertions, deletions, and substitutions.

7.1 Object Detection and Gold N-grams

After subjecting the output to the machine learned patterns, the Levenshtein distance shrunk from 280 to 261. Both the Random Forest (RF) and Sliding Window (SW) approaches, improved the performance, using either the gold standard N-grams and those that were derived from the OMR tool. As can be observed in Table 1.

7.2 Patterns

The patterns derived from gold are displayed in Table 2, while those derived from the OMR output are displayed in Table 3. The pattern matching is shown in Table 4, here the missing note was replaced with the number 1000. A Levenshtein distance of exactly one, therefore generates suggestions to fill in this missing note. The patterns and frequencies differ a lot between the two, mainly

[1]https://github.com/Tomeriko96/Optical-Music-Recognition-Tool

Table 1: Levenshtein distances to gold standard.

Method	Levenshtein	Precision
OMR	280	80%
Gold RF	263	81%
OMR RF	258	81%
Gold SW	260	81%
OMR SW	260	81%

Table 2: Patterns from gold standard.

Pattern	Frequency
-2.0, -2.0, 4.0, -2.0	10
-2.0, -2.0, 4.0, -8.0	9
-2.0, 4.0, -8.0, 0.0	8
-2.0, -1.0, 3.0, -7.0	7
-2.0, -2.0, -2.0, 4.0	7
0.0, -2.0, -2.0, 4.0	7
-2.0, 4.0, -2.0, -2.0	7
-1.0, 3.0, -7.0, 0.0	6
0.0, 0.0, -2.0, -2.0	6
0.0, -2.0, -2.0, -2.0	5

Table 3: Patterns from OMR output.

Pattern	Frequency
0.0, 0.0, 0.0, 0.0	18
-3.0, 0.0, 0.0, 0.0	4
-1.0, 4.0, -8.0, 0.0	3
0.0, 0.0, 1.0, 0.0	3
1.0, 3.0, -4.0, 1.0	3
3.0, -4.0, 1.0, 3.0	3
0.0, 1.0, -1.0, 1.0	3
0.0, 2.0, 0.0, 0.0	3
1.0, 0.0, 0.0, 0.0	3
0.0, 0.0, 0.0, -1.0	3

Table 4: Examples of pattern matching in the SW approach.

Missing Pattern	Suggestion
0.0, 0.0, **1000**, 1.0	0.0, 0.0, **-1.0**, 1.0
4.0, -8.0, 0.0, **1000**	4.0, -8.0, 0.0, **6.0**
0.0, **1000**, -2.0, -2.0	0.0, **0.0**, -2.0, -2.0

because of the missing notes and wrong classifications in the OMR phase. Also a lot more zero's are included in the patterns of the OMR tool. These zero's occur mostly after a missing note was preceded. This would also explain the lower frequencies, due to repeating patterns being interrupted by missing notes and having to start over at zero. Nevertheless, the same improvement was achieved by the patterns from the imperfect OMR tool, as from the gold standard patterns.

8 DISCUSSION

One of the main caveats, of this project, was the lack of a suitable dataset. Although building it was a valuable learning experience, it might contain errors, due to a more limited musical knowledge at the time of labeling. This can be expressed, for instance, by looking at the labels of the different representations of the rests. In hindsight, knowing that each rest is coupled to its own note, labelling could have been more straightforward. Especially after finding out that the tinynotation module visualizes the rests automatically based on the note length.

Furthermore, the symbol labelling occurred weeks before the note head detection was even thought of. This lead to the unnecessary labelling of the two different conformation of each note. Originally the position of the note head was to be derived from the bounding boxes generated in the first object detection. However, this proved inconsistent as some boxes were slightly off, leading to inconsistent note head positions. Therefore, the division in a note with the head pointed upwards or downwards was, in hindsight, a poor labelling choice which also slowed down labelling and required additional steps in the data processing.

Another limitation with regards to the dataset, is its small size. As mentioned earlier, the dataset was built using three music scores, all from the same piece. In order to have more varied data, a larger corpora should have been used for labelling.

However, having a larger corpus, with for instance varying composer, might have rendered certain N-grams useless, as it can not be sure whether the different composer would have implemented similar patterns.

Furthermore, the limits of the tinynotation module of music21, lead to minor inconsistencies in transcribing the images. With regards to note lengths, we were unable to display breves as being double whole notes. The module allows user to append the following values to notes: 1 for a whole note, 2 for half a note, 4 for a quarter, 8 for an eight, and 16 for a sixteenth note. Furthermore, issues occurred while constructing the modern output. After hours of playing around with the music21 module, we were unable to figure out how to display notes in certain clefs, using the tinynotation module. This issue forced the output to be in the wrong clef, and therefore not be accurate. Due to these two critical limitations in the Tinynotation module, it was decided to manually append information to a stream, instead of using Tinynotation, in the case of certain clefs. The output image therefore became less visually appealing, but the output was more accurate this way.

Due to the limited amount of gold standard data available, this work only presents the findings on a small piece, written in one clef only. In order to validate the findings in this paper, future work should look into applying this framework on a larger corpus, containing a variety of scores.

The degraded conditions of the original images have caused the method to make mistakes. This has lead to missed symbols and mis-classifications. In particular smaller details, such as note heads and dots were either missed or mistaken. The conditions also caused the algorithm to find less than five lines. A very small amount of images caused this undesirable behaviour, even after the preprocessing stage was repeated.

Also, due to the inclusion of lyrics and capital letters, the Hough transform algorithm tended to make mistakes in line detection. The rough lines present in the mentioned object were therefore mistakes for lines, which threw off the detection. This in cause lead to the wrong localization of note heads and incorrect translations. Therefore, for this application, the assumption holds that images need to be cropped in such a way that no noise is present, such as the aforementioned lyrics and capital letters. Nevertheless, segmentation the staves from an image would be a useful feature to have. When implemented, an entire page could be parsed and be transcribed automatically, without the need to manually extract the staves.

As the object detection is far from perfect, some notes are missing in the final representation. This also affects the N-grams as some might have gone missing due to the mentioned limitation.

As can be seen from the images, some of the symbols detected have a low percentage of confidence. This is mainly caused by the low amount of training done on the data. If the training would have been run for 40.000 epochs instead for the 10.000 epochs done here, it might return suggestions with higher confidence, and also identify those symbols it currently did not identify. Chosen was ultimately to keep it at 10.000 epochs as training took very long on the computer used.

Given that the second object detection fails to find a note head, the note value must be derived from the first object detection. The problem is that the bounding box encompasses the entire note, therefore making it hard to get an accurate location of the note head. While patterns do consist of symbols only, they are not independent from the metadata, such as the time signature such as rhythm and pitch. In this project, the metadata was not taken into account. However, future research may benefit from using such data. Another limitation is the interference of the rests. Therefore, the patterns derived may not be correct, as rests should have been included.

Looking back at the chosen method of evaluation, Hamming distance would have been the preferred method of evaluation. However, the algorithm requires that the input strings are of the same length [9]. The tool built in this paper, is not perfect in detection and, as a result, occasionally misses notes. Therefore, Levenshtein distance became the method of evaluation, foregoing the requirement of equal lengths.

Finally, the symbol detection model was, in fact, never trained on instances written in the Tenor clef, as the training set did not consist of any pieces written in that clef. Due to similarities in appearance, the Alto clef was therefore detected, which was part of the training data.

9 CONCLUSION

In this paper, we set out to improve improve upon existing OMR systems using knowledge derived from N-grams. These patterns were derived once from a gold standard, and the other time from the output of the OMR tool. These patterns were then used to test two approaches. The first approach consisted of the unsupervised machine learning algorithm was trained on the N-grams. The other method took these patterns and used a sliding window matching algorithm. Both approaches improved the prediction, both from the gold standard as from the predicted output. Therefore, all research question can be answered positively.

Given our limited amount of data to work with, future work should focus on applying this approach on a larger corpus to validate the results of this paper. The next step in this research is to collect as much data as possible, preferably from scores originating from various time periods, in order to fully exploit the potential of the pattern recognition approach. Also, while this work only focused on the use of pattern matching in the task of filling missing notes, future work should look into the possibility of substitutions and deletions. Lastly, to get the complete picture of the power of the N-grams, future work should take into consideration permutations of patterns. The original pattern would then be considered the prime pattern, its mirrored version on the y axis, the retrograde. Conversely, the translation on the x-axis of the prime pattern would be the inverse, and the translation of the inverse on the y-axis, the retrograde-inverse. Recapitulating, with the current trends in OMR and pattern recognition, it should be possible to get closer to an application that can perfectly translate White Mensural Notation without human intervention.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Willi Apel. 1961. *The notation of polyphonic music, 900-1600.* Number 38. Medieval Academy of Amer.

[3] Pierfrancesco Bellini, Ivan Bruno, and Paolo Nesi. 2001. Optical music sheet segmentation. In *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001.* IEEE, 183–190.

[4] Dorothea Blostein and Henry S Baird. 1992. A critical survey of music image analysis. In *Structured Document Image Analysis*. Springer, 405–434.

[5] Michael Scott Cuthbert and Christopher Ariza. 2010. Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data.. In *ISMIR*, J. Stephen Downie and Remco C. Veltkamp (Eds.). International Society for Music Information Retrieval, 637–642. http://dblp.uni-trier.de/db/conf/ismir/ismir2010.html#CuthbertA10

[6] R Demopoulos and Michael J Katchabaw. 2007. Music information retrieval: a survey of issues and approaches. *Technical Report* (2007).

[7] Michael Droettboom, Karl MacMillan, and Ichiro Fujinaga. 2003. The Gamera framework for building custom recognition systems. (2003).

[8] Richard O Duda and Peter E Hart. 1971. *Use of the Hough transformation to detect lines and curves in pictures.* Technical Report. SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER.

[9] Richard W Hamming. 1950. Error detecting and error correcting codes. *The Bell system technical journal* 29, 2 (1950), 147–160.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision.* 2961–2969.

[11] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[12] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 51 – 56.

[13] Nobuyuki Otsu. 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics* 9, 1 (1979), 62–66.

[14] Alexander Pacha, Jan Hajič, and Jorge Calvo-Zaragoza. 2018. A Baseline for General Music Object Detection with Deep Learning. *Applied Sciences* 8, 9 (2018), 1488.

[15] Emilia Parada-Cabaleiro, Anton Batliner, Alice Baird, and Björn Schuller. 2017. The SEILS dataset: Symbolically encoded scores in modern-early notation for computational musicology. (2017).

[16] Jeremy Pickens. 2001. A Survey of Feature Selection Techniques for Music Information Retrieval.

[17] Laurent Pugin. 2006. Optical Music Recognitoin of Early Typographic Prints using Hidden Markov Models.. In *ISMIR*. 53–56.

[18] Laurent Pugin, Jason Hockman, John Ashley Burgoyne, and Ichiro Fujinaga. 2008. GAMERA versus ARUSPIX–TWO OPTICAL MUSIC RECOGNITION APPROACHES. In *ISMIR 2008–Session 3C–OMR, Alignment and Annotation*. Citeseer.

[19] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre RS Marcal, Carlos Guedes, and Jaime S Cardoso. 2012. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval* 1, 3 (2012), 173–190.

[20] David Rizo, Jorge Calvo-Zaragoza, and José M Iñesta. 2018. MuRET: a music recognition, encoding, and transcription tool. In *Proceedings of the 5th International Conference on Digital Libraries for Musicology*. ACM, 52–56.

[21] Lorenzo J. Tardón, Simone Sammartino, Isabel Barbancho, Verónica Gómez, and Antonio Oliver. 2009. Optical Music Recognition for Scores Written in White Mensural Notation. *EURASIP Journal on Image and Video Processing* 2009, 1 (28 Dec 2009), 843401. https://doi.org/10.1155/2009/843401