USING SEQUENCE ALIGNMENT AND VOTING TO IMPROVE OPTICAL MUSIC RECOGNITION FROM MULTIPLE RECOGNIZERS

Esben Paul Bugge Kim Lundsteen Juncher Brian Søborg Mathiasen Jakob Grue Simonsen
Department of Computer Science, University of Copenhagen (DIKU)
Njalsgade 126–128, 2300 Copenhagen S, Denmark
{ebugge,juncher,soborg,simonsen}@diku.dk

ABSTRACT

Digitalizing sheet music using Optical Music Recognition (OMR) is error-prone, especially when using noisy images created from scanned prints. Inspired by DNA-sequence alignment, we devise a method to use multiple sequence alignment to automatically compare output from multiple third party OMR tools and perform automatic error-correction of pitch and duration of notes.

We perform tests on a corpus of 49 one-page scores of varying quality. Our method on average reduces the amount of errors from an ensemble of 4 commercial OMR tools. The method achieves, on average, fewer errors than each recognizer by itself, but statistical tests show that it is significantly better than only 2 of the 4 commercial recognizers. The results suggest that recognizers may be improved somewhat by sequence alignment and voting, but that more elaborate methods may be needed to obtain substantial improvements.

All software, scanned music data used for testing, and experiment protocols are open source and available at:
`http://code.google.com/p/omr-errorcorrection/`

1. INTRODUCTION AND RELATED WORK

Optical music recognition (OMR) is an active field, but suffers from a number of technical pitfalls, even in the "typical" case where only music notation in modern, conventional western style is considered [3,8,13]. While affordable commercial tools for OMR are available, imperfections in scanned sheet music make these error-prone (see Fig. 1).

One possibility for improving the accuracy of OMR programs is to use *multiple recognizers*: Let several programs (*recognizers*) perform OMR independepently, and combine the results afterwards using a *combined recognizer*. The practical possibility of using multiple recognizers has been investigated by Byrd et al. [5–7], and appears promising, but brings new pitfalls with it; in extreme cases, OMR programs could fail dismally at different tasks, hence–in theory–making the combined result *worse* than the output of the individual recognizer.

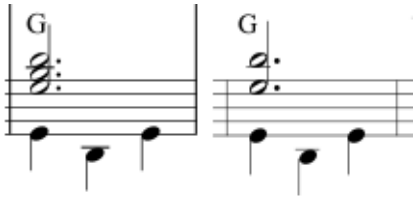


Figure 1. Example of a recognizer missing a note. Left: Bar 6 of the bass part of a piano arrangement of "God save the Queen" by T.A. Arne. Right: The output of Capella-Scan 1.6.

In contrast, we take a workmanlike approach to multiple recognizers: The basic tenet is that every commercially available tool will not fail dismally on a single aspect of OMR in *most* cases (the product would be too poor to use), and that different tools are likely to fail in different aspects. Byrd et al. [6,7] suggest amassing a set of rules, or attaching weights to certain single recognizers, based on their prior performance, to obtain maximal increase of accuracy in a multi-recognizer tool; however, they also note that this is a moving target, due to new versions of existing products improving on some aspect of recognition. In contrast, we are simply satisfied if a multi-recognizer *is, on average, better than any single-recognizer, to a high degree of statistical significance*.

To account for the fact that different recognizers may make different errors, hence causing misalignment of their respective outputs (see Fig. 2) we align their outputs using a multiple sequence alignment algorithm and subsequently use a simple voting procedure to resolve conflicts. A prerequisite for such an approach to work is that no single recognizer significantly outperforms the others, as a multiple recognizer would then perform *worse* as the suboptimal rec-

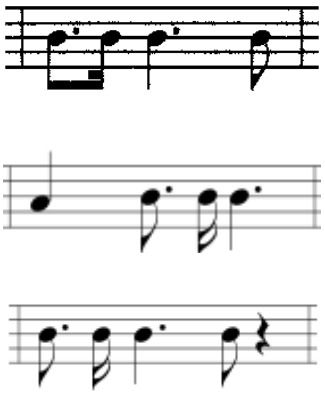


Figure 2. Misaligned notes from the third bar of "Mon beau sapin" by E. Anschütz. Top: Original; middle: Capella-Scan 6.1; bottom: Photoscore Ultimate 6.

ognizers introduce noise in the sequence alignment.

Our work was originally motivated by our desire to examine melodic and harmonic progression as used by different composers, and how the statistical properties of such progressions changed over the lifetime of composers. Our results are thus restricted to aspects of melody and harmony; we thus consider only notes, rests, bars, keys, etc., but omit dynamic indications (p, pp, etc.) and the–admittedly more difficult–problem of slurs and complex annotations.

1.1 Related work

Byrd et al. [5–7] report on several experiments using an OMR system based on several different recognizers, including a prototype system for sequence alignment, but do not give details on the numerical improvement of the multi-recognizer system. Szwoch [15] uses alignment within bars to automatically obtain error counts for OMR systems, but does not give numerical evidence. Pardo and Sanghi [12] employ multiple sequence alignment to find optimal matching works in databases of polyphonic music when queried with monophonic pieces; they consider an alphabet where each musical symbol is a note with pitch and duration, and each part in a polyphonic score corresponds to a sequence. Allali et al. substantially extend this approach to encompass polyphonic queries [1, 2].

While the work of Byrd et al. is very similar to ours, we believe our work offers the following incremental benefits: (i) confirmation of the positive results obtained in the experiments of Byrd et al., (ii) comparison of different commercial tools with each other and with a system based on multiple recognizers with statistical significance testing, (iii) full, numerical reporting of results, (iv) full release of all tools as open-source software, including the MusicXiMpLe XML Schema Definition (XSD) and sequence alignment software.

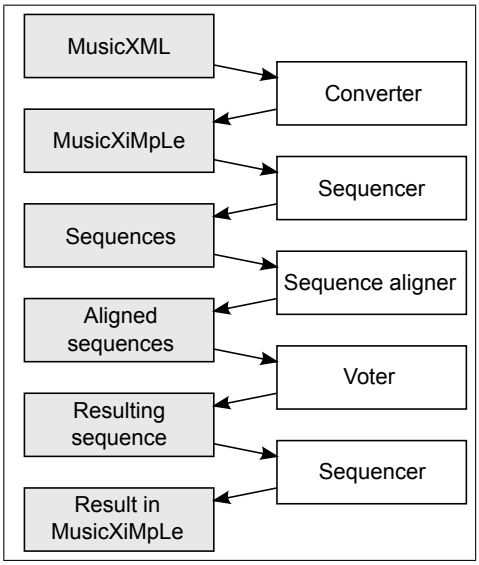


Figure 3. Pipeline for the OMR system. Rectangles represent data objects and boxes machinery for processing or converting data. The left topmost rectangle contains n different pieces of MusicXML data from n different OMR programs.

2. ALIGNMENT OF OUTPUT FROM MULTIPLE RECOGNIZERS: PRACTICAL OVERVIEW

Our combined recognizer takes the output from several recognizers in a common format, converts the output to several sequences of musical symbols which are then aligned with conflicts resolved by majority (colloquially: "The programs vote for the symbols" after alignment); the resulting sequence is then converted to the common format (see Fig. 3.

We employed four commercial recognizers: Capella-Scan 6.1, SmartScore X Pro 10.2.6, PhotoScore Ultimate 6, and SharpEye 2. VivaldiScan was briefly investigated, but discarded as it (for our purposes) was only a wrapper for the OMR procedures of SharpEye. All tools support several output formats; we chose MusicXML as all programs support it and the format is amenable to manipulation.

The *converter* converts MusicXML to a standard format called MusicXiMpLe (see Section 2.1) with the purpose of normalizing notation. The *sequencer* converts MusicXiMpLe to an internal representation of music as a sequence of symbols (see Section 3.1). The *Sequence aligner* (see Section 3.2) uses multiple sequence alignment to align the sequences, and the *Voter* is used to settle disputes among OMR programs after alignment. The *Sequencer* is then used again to convert from the internal sequence representation to the standard format.

2.1 A common output format: MusicXiMpLe

Due to the ambiguities in MusicXML, a piece of music can be represented in different ways, and different recognizers may output starkly different MusicXML, even if all recognizers read the music correctly. Furthermore, MusicXML is quite verbose, containing more information and metadata than needed for our experiment. To address these issues, we created an XML Schema Definition (XSD) containing solely those elements needed for analysis. We call the set of XML-data conforming to our XSD "MusicXiMpLe"; note that valid MusicXiMpLe is also valid MusicXML.

Briefly, MusicXiMpLe holds the following data. In contrast to ordinary MusicXML, restrictions are noted in [square brackets]: (i) parts [each part holds *exactly* one staff], (ii) measures [only part-wise structures are allowed, not time-wise], (iii) notes [only pitch, duration, octave, alternation and simultaneity are recorded], (iv) rests [only duration is recorded], (v) the MusicXML "musical counter", (vi) repeats and alternative endings, (vii) time-signature, (viii) key, (ix) chord symbols.

3. MUSICAL SYMBOLS AND MUSIC DATA AS A SEQUENCE

Sequence alignment is the task of comparing and aligning $n > 1$ sequences of symbols. As an example, consider the sequences s_1 and s_2 constructed using the symbol set {`A`, `B`, `C`, `D`, `E`}:

$$\begin{aligned} s_1 &= \texttt{AABBCCDA} \\ s_2 &= \texttt{ABCE} \end{aligned}$$

Sequence alignment of s_1 and s_2 might give the following result (depending on the algorithm used):

$$\begin{aligned} a_1 &= \texttt{AABBCCDA} \\ a_2 &= \texttt{-AB-CE--} \end{aligned}$$

where a_1 and a_2 represents the aligned sequences of s_1 and s_2 respectively and '$-$' represents a gap inserted by the alignment algorithm.

Sequence alignment algorithms calculate similarity scores for the elements in the sequences; high similarity will occur at points in a score where two recognizers output the same symbols, for instance barlines in the same place. These scores are then used to align the sequences. When given N sequences as input, multiple sequence alignment returns N aligned sequences, possibly with gaps inserted. In our case, this corresponds to N aligned scores; we will reduce these to a single score, by letting each recognizers "vote" for each single element in the N aligned sequences (ties broken randomly).

3.1 Symbolic music data as a sequence

We consider music data as any sequence of elements e where e is generated from the following grammar:

$$e := \text{note}^+ \mid \text{rest} \mid \text{barline} \mid \text{repeat} \mid \text{ending} \mid \text{key} \mid \text{time} \mid \text{clef}$$

A note above is a quadruple (p, a, o, l) where $p \in \{A, \ldots, G\}$ is the *pitch class*, $a \in \{\text{flat}, \text{natural}, \text{sharp}\}$ the *alternation*, $o \in \{0, \ldots, 9\}$ the *octave*, and $l \in \mathbb{Q}$ the *duration* of the note. An element holds one or more notes, hence may function as a chord. The notes in an element may have different lengths (see Fig. 4). Intuitively, the sequence has an element for each "change" in the music. With chords containing notes of different lengths, a single note missed by a recognizer may lead to very distinct sequences of elements for two different recognizers (this problem is addressed in the sequence alignment, as similarity scores between elements are computed in such a way that elements that only differ by "few" notes are counted "almost similar").

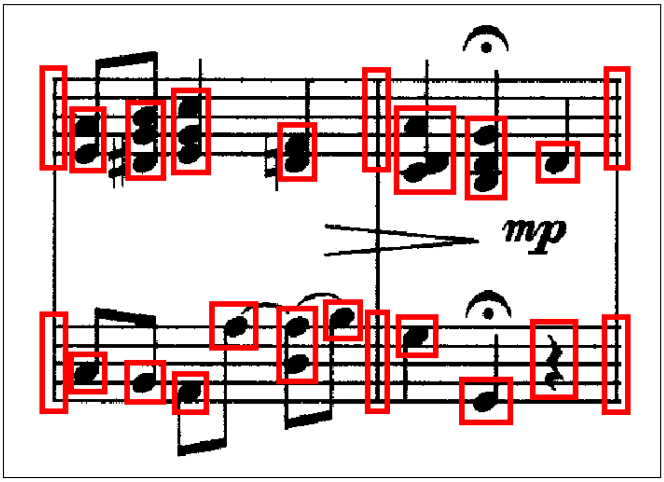


Figure 4. Bars 11–12 of "O Christmas tree!" by E. Anschütz . Elements of the sequence alphabet are indicated by red outlines (accidentals and duration are included in each element).

We consider each staff to hold a single sequence of symbols, and perform sequence alignment per-staff. For sheet music with notes where it is unclear to which staff a given note belongs, different recognizers may assign notes to different staves, negatively affecting subsequent sequence alignment.

3.2 Progressive sequence alignment of symbolic music data and voting

We briefly outline the method for multiple sequence alignment below. Note that our choice of algorithms is not due to any intrinsic properties of symbolic music; the employed

algorithms could very likely be replaced by other algorithms from the sequence alignment literature without detrimental effect to correctness or performance.

Due to its tradeoff between speed and precision, we employ *progressive* multiple sequence alignment [16] in which (a) pair-wise alignment of all sequence-pairs is performed, followed by (b) computation of a similarity-score D for each pair, and (c) the two most similar are aligned first, producing two new sequences that are then (d) progressively aligned with the remaining sequences in descending order of similarity score.

Progressive alignment is greedy and non-optimal–as opposed to dynamic programming methods–but is significantly faster. For pairwise alignment, we use the classic *Needleman-Wunsch algorithm* [11]. This method finds the alignment of two sequences s_1 and s_2 of length k and l by first creating the *similarity matrix* M defined by the $(k+1, l+1)$-dimensional matrix $M_{i,j}$ where $M_{0,j} = g \cdot j$, and

$$M_{i,j} = \max \left| \begin{array}{l} M_{i-1,j-1} + \alpha(s_1[i], s_2[j]) \\ M_{i-1,j} + g \\ M_{i,j-1} + g \end{array} \right. \quad (1)$$

where $i \in \{0, 1, ..., k\}$, $j \in \{0, 1, ..., l\}$, the function $\alpha(x, y)$ returns a score based on whether the two elements x and y are similar or not, and g is the *gap penalty* which is the score of inserting a gap into one of the sequences. In addition, the algorithm maintains a *trace matrix* T of identical dimensions. This matrix holds information about how the value of each element in M was found. If for example the value of $M_{1,2}$ is $M_{0,1} + \alpha(s_1[1], s_2[2])$, $T_{1,2}$ will hold the coordinates (0,1).

For two musical elements e_1, e_2, we define their similarity as $\alpha(e_1, e_2) = d$ if the elements are completely distinct, and $\alpha(e_1, e_2) = ks/n$ if the elements are similar, where n is the combined number of symbols in e_1 and e_2, and k is the number of symbols they have in common (note that notes of identical pitch, but different length are counted as being distinct). The parameters g, d and s can be set according to preference or performance. All our experiments were conducted with $g = -2$, $d = -1$ and $s = 1$. To avoid spurious "elements" containing notes in combinations with time signatures, bar lines or clefs, such combinations were heavily penalized by setting their similarity scores effectively to $-\infty$.

When the matrices M and T have been constructed, pairwise alignment proceeds by following the path from $T_{k,l}$ back to $T_{0,0}$ using the coordinates stored in the cells of T. In the example above, the returned solution is $a_1 =$ `ABBCE`, $a_2 =$ `A--CD`.

To extend the pairwise alignment to *multiple* alignent, a so-called *guide* is constructed that specifies the sequence in which pairwise alignments are performed. The guide is constructed by the standard technique of *neighbor-joining* [14].

		A	B	B	C	E
	0	-2	-4	-6	-8	-10
A	-2	1	-1	-3	-5	-7
C	-4	-1	0	-2	-2	-4
D	-6	-3	-2	-1	-3	-3

Figure 5. A similarity matrix M using input strings $s_1 =$ABBCE and $s_2 =$ACD.

		A	B	B	C	E
		(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
A	(0,0)	**(0,0)**	**(1,1)**	**(1,2)**	(1,3)	(1,4)
C	(1,0)	(1,1)	(1,1)	(1,2) (2,2)	**(1,3)**	(2,4)
D	(2,0)	(2,1)	(2,1) (2,2)	(2,2)	(2,3) (3,3)	**(2,4)**

Figure 6. The trace matrix T corresponding to the similarity matrix from Figure 5. Entry $T_{i,j}$ holds the coordinates of the entry that led to the value of the lower-right entry of M. Multiple coordinates in an entry give rise to multiple paths. The path corresponding to the optimal solution is highlighted in bold: ((2,4) → (1,3) → (1,2) → (1,1) → (0,0)).

For every position in the set of N aligned sequences, we collect all symbols from all sequences and their count. For a symbol to be included in the final output, it must have an absolute majority (exceptions are clefs and time signatures that only need half the votes, as we found that the existing recognizers tend to miss them).

4. EXPERIMENT

We collected a corpus (*Corpus A*) of 25 scanned, public domain, one-page pieces of western classical music. The corpus consisted solely of western classical music ranked in 5 groups of 5 each according to quality (1 worst, 5 best; see Fig. 7). The corpus was composed prior to any OMR scanning by the various recognizers; the music ranged from 1–15 staves with either chords or multiple voices present in most staves. We supplemented Corpus A by acquiring the 24 scanned pages from the original study of Byrd et al. [6] (*Corpus B*). This corpus consisted mostly of high-quality scans (qualities 3–5 on our scale) with mostly a single voice on a single staff. In both corpora, we employed 300DPI scans, using the "uncleaned" scans of Corpus B. We applied the four commercial products to the combined corpus A+B, using Finale Songwriter 2010 to read the output MusicXML, and performed error counts by hand.

4.1 Error counting

Error counting in OMR is notoriously difficult and ambiguous [3, 4, 6, 9]. Droettboom and Fujinaga [9] and Bellini et al. [4] argue that error counting at the level of atomic symbols such as noteheads, flags etc. is markedly different from the case with composite symbols (beamed notes, chords, etc.), and that a single error in an atomic symbol may cause

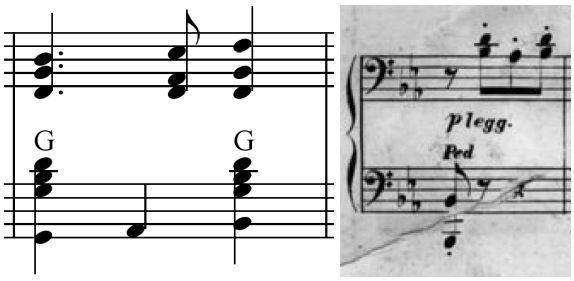


Figure 7. Scores of quality 5 (left: Bar 12 of "God save the Queen" by T.A. Arne) and quality 1 (right: Staff 3, Bar 17 of "La Baladine Caprice" by C.B. Lysberg).

numerous errors in composite symbols out of proportion. In addition, there are inherent ambiguities in error counting (see Tab. 1). There appears to be no consensus in the literature on the "correct" way to resolve ambiguities, so we chose as a guideline that the *sound* (pitch, duration, etc.) of the music should be preserved, that is, if a recognizer fails to read symbols correctly, but replaces them with identically sounding ones (e.g. replacing a whole note rest by two half note rests), we do *not* count it as an error. However, to avoid penalizing OMR tools for missing the beginning clef or key signature (in which case most or all of the notes in the piece would be counted as in error), we only count *one* error for such a miss. For potential ambiguities in the error count, we followed a strict disambiguation procedure, described in Table 1 along with their resolution.

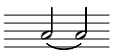
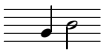
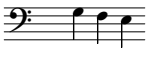
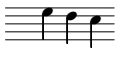
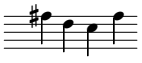
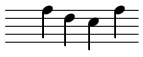
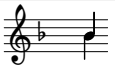
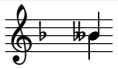
Original score	Post-OMR score	Ambiguity (A) and Resolution (R)
		A: Unclear which of the two notes is missing. R: Count one `note missing` error.
		A: Note has been misread both in duration and pitch. R: Counts as one note error.
		A: Unclear which note is missing and which note has been transposed. R: Count one missing note and one transformed note, yielding two errors.
		A: Unclear which of three notes is missing. R: Count one missing note and one transformed note, yielding two errors.
		A: Unclear how the remaining notes after missing clef should be read. R: Count one missing clef, no note errors, yielding one error.
		A: Unclear of the effect of the missing sharp pitch. R: Missing accidentals results in note errors for every alterated note within the tab, yielding two errors.
		A: Unclear how to count the added accidentals. R: The MusicXiMpLe format adds the extra accidentals, and these are denoted for each note. This yields no errors
		A: The resulting document from conversion to MusicXiMpLe breaks beams. R: Cosmetic issue, yields no errors.

Table 1. (Non-exhaustive) list of common ambiguities for error counts and their resolution

4.2 Qualitative assessment

Naked-eye inspection during error counts revealed that all recognizers have errors on most pages. Furthermore, the combined recognizer seems to perform better on Corpus B than on Corpus A, containing mostly single-staff, single-voice music. It would thus appear that sequence alignment and voting is impaired by chords, and that a refined distance metric between "similar" chords is needed. Another opportunity for improvement is that the sequence alignment is affected negatively if several recognizers misread a clef: All notes will be dissimilar, to the detriment of the alignment algorithm; this problem could possibly be avoided by letting each recognizer output using a *notation* format or relative pitch notation, rather than a music format (where pitches are absolute).

4.3 Quantitative assessment

For testing whether one recognizer significantly outperformed the other, we performed an experiment with our two corpora ($N = 49$). To avoid spurious assumptions about the normality of the error rate of each recognizer, we eschewed parametric tests and instead performed (a) non-parametric Friedman tests on the ensemble of all tools, (b) sign tests on each pair of recognizers against the null hypothesis that applying a pair of recognizers to a random score the recognizers are equally likely to yield fewer errors than the other. Both tests avoid debatable comparisons of the absolute number of errors per page, comparing only the relative number of errors for each pair of recognizers. Tests were performed at significance level of $p < .05$.

Ranking the five recognizers from least errors (rank 1) to most errors (rank 5), the combined recognizer (CR) performed best on average: CR: 2.43, Sharpeye: 2.83, Smartscore: 2.86, Photoscore: 3.26, Capella-Scan: 3.62. The Friedman test showed a significant difference in the set of ranks of the five recognizers ($\chi^2 = 16.286$, df = 4, $p = .003$). A post-hoc sign test with Bonferroni correction only yielded significance for the pair CR vs. Capella-Scan ($Z = -3.166$, $p < .005$). The sign test on all pairs of recognizers yielded significant results for CR vs. Photoscore ($Z = -1.960$, $p = .049$), CR vs. Capella ($Z = -3.166$, $p = .001$), and Capella-Scan vs. Sharpeye ($Z = -2.261$, $p = .023$), while the remaining pairwise comparisons were non-significant.

The results suggest that Capella-Scan often made more errors than the remaining tools, and that Sharpeye often made fewer errors. The sign test also revealed that none of the recognizers *consistently* outperform each other, for example in the 46 scores that both recognizers were able to scan, Capella-Scan had fewer errors than Sharpeye in 14, 2 ties, and more errors in 30.

While the average rankings of the tools suggest that the

combined recognizer generally performs better, the fact that we can only give reasonable statistical evidence for this supposition for two of the commercial tools tempers the conclusion somewhat. We have little doubt that given a test corpus of scores in the hundreds we would obtain significant differences for the remaining tools, but clearly the improvement is small. Even for high-quality (4 and 5) scores, all recognizers had error counts above 0 (only on a single score in Corpus B did every tool perform spotlessly). It appears that fully-automated, error-free music recognition is not possible and that human post-correction is almost invariably warranted.

5. CONCLUSION AND FUTURE WORK

We have shown that a simple OMR system based on multiple recognizers and sequence alignment can outperform the commercially available tools. Our results confirm the earlier work of Byrd et al. suggesting that recognizers may be improved somewhat by sequence alignment and voting, but that more elaborate methods may be needed to obtain substantial improvements. For future work, we suggest tackling dynamics, slurs, articulations, ornaments, arpeggiated chords, and other embellishments. We advocate the establishment of sizable online repositories of scores both for benchmarking multiple recognizers *and* for the output of such systems (i.e., *reliable, error-free* scores), using a suitable interchange format, e.g. [10].

6. REFERENCES

[1] Julien Allali, Pascal Ferraro, Pierre Hanna, Costas S. Iliopoulos, and Matthias Robine. Toward a general framework for polyphonic comparison. *Fundam. Inform.*, 97(3):331–346, 2009.

[2] Julien Allali, Pascal Ferraro, Pierre Hanna, and Matthias Robine. Polyphonic alignment algorithms for symbolic music retrieval. In *CMMR/ICAD*, volume 5954 of *Lecture Notes in Computer Science*, pages 466–482. Springer, 2009.

[3] David Bainbridge and Tim Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.

[4] P. Bellinni, I. Bruno, and P. Nesi. Assessing optical music recognition tools. *Computer Music Journal*, 31(1):68–93, 2007.

[5] David Byrd and Ian Knopke. Towards musicdiff: A foundation for improved optical recognition using multiple recognizers. In *Proceedings of ISMIR '07*, pages 123–126, 2007.

[6] Donald Byrd, William Guerin, Megan Schindele, and Ian Knopke. Omr evaluation and prospects for improved OMR via multiple recognizers. Submitted for publication.

[7] Donald Byrd and Megan Schindele. Prospects for improving OMR with multiple recognizers. In *Proceedings of ISMIR '06*, pages 41–46, 2006.

[8] Jaime S. Cardoso and Ana Rebelo. Robust staffline thickness and distance estimation in binary and gray-level music scores. In *CPR*, pages 1856–1859. IEEE, 2010.

[9] Michael Droettboom and Ichiro Fujinaga. Micro-level groundtruthing environment for OMR. In *ISMIR*, 2004.

[10] Andrew Hankinson, Laurent Pugin, and Ichiro Fujinaga. An interchange format for optical music recognition applications. In *Proceedings of ISMIR '10*, pages 51–56, 2010.

[11] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, March 1970.

[12] Bryan Pardo and Manan Sanghi. Polyphonic musical sequence alignment for database search. In *Proceedings of ISMIR '05*, pages 215–222, 2005.

[13] Ana Rebelo, G. Capela, and Jaime S. Cardoso. Optical recognition of music symbols - a comparative study. *International Journal of Document Analysis and Recognition*, 13(1):19–31, 2010.

[14] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[15] Mariusz Szwoch. Using musicxml to evaluate accuracy of OMR systems. In *Diagrams*, volume 5223 of *Lecture Notes in Computer Science*, pages 419–422. Springer, 2008.

[16] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.