MELODY RETRIEVAL AND COMPOSER ATTRIBUTION USING SEQUENCE ALIGNMENT ON RISM INCIPITS

Jelmer van Nuss
Utrecht University
j.l.vannuss@students.uu.nl

Geert-Jan Giezeman
Utrecht University
G.J.Giezeman@uu.nl

Frans Wiering
Utrecht University
F.Wiering@uu.nl

ABSTRACT

The RISM A/II database contains metadata and incipits of more than a million compositions. The Monochord search engine can retrieve incipits that are similar to a query using several alignment methods based on pitch raters, weight-based raters and duration-based raters. The performance of all 27 search methods is evaluated using Mean Average Precision metrics and the TREC framework for retrieval performance analysis. The difference in exact pitch between melodies turns out to be the best factor to search with for musical similarity retrieval.

All melodies have metadata such as a composer name, but a portion of the database is labelled as *Anonymus*. A k-Nearest Neighbours algorithm is optimised for the purpose of deanonymisation and used to classify several *Anonymus* songs to test the applicability of this classifier for composer labelling. Using a classifier as a first selection step for deanonymisation purposes turns out to be viable with human correction.

1. INTRODUCTION

The RISM A/II database is a collection of melodies that are stored as incipits, excerpts from the beginnings of notated music in manuscripts collected from libraries, archives, monasteries and schools [1, 2]. This database is not only a useful tool for information look-up on one song, but also to collect similar melodies that give a broader context of the researched melody. As over a million melodies have been stored in this database, having an effective search engine is crucial. While the RISM website [1] has a search function for both metadata and music notation, the power of the search method is limited since it cannot take musical similarity into account. This is why an alternative search engine, Monochord [2], employing more advanced and possibly more accurate search techniques, has been developed.

Monochord is a music retrieval system that is able to find melodies in the RISM A/II database [3]. Monochord compares two melodies by aligning them and calculating a similarity score. The higher this score is, the better the match between the melodies. This search engine has several retrieval methods at its disposal. The performances of these methods have not been researched previously, yet these have to be known before any claims about the performance relative to RISM's search engine can be made.

The goal of this research is twofold. Our first problem is finding an optimal combination of settings that finds similar melodies. Second, after having determined this combination of settings, we use it in an experiment to deanonymise a part of the RISM database.

We begin with an evaluation of the search methods provided by the Monochord search engine. In testing the workings of this engine, we not only gain better understanding of the capabilities and limits of the music similarity retriever, but also gain insights in how to improve the search methods. This part is modelled on previous research conducted by Typke [2], who used similar techniques to create a ground truth set and to evaluate the retrieval results.

Of the 1.148.478 melodies currently stored in the Monochord database, 214.162 have an unknown composer: these are labelled *Anonymus*. Some of these melodies might actually be composed by a composer whose name is impossible for us to retrieve. Others are similar to melodies of which the composer is known. A third type of *Anonymus* songs is that of traditional material that has no single apparent composer. It is desirable to know the true composer of a melody to give credit to the musician, but also to place the works in their context, which may lead to new insights in music history. Using the metadata of similar melodies, we create a classification procedure to determine the composer of the anonymous incipits.

This experiment has a preparation phase and an analytical phase. First off, it is important to understand the mechanisms behind Monochord, how it uses alignment of melodies and several raters to determine melodic similarity, and which aspects differ from the RISM search engine [3]. The retrieval results have to be compared to a ground truth set, which is an expanded version of the one created by Typke [2]. This comparison is made based on precision-recall curves created by means of standard retrieval evaluation tools. The Monochord engine resembles the top k selection used by a k-Nearest Neighbours algorithm. A k-NN model is thus prepared to suggest composer labels for *Anonymus* melodies.

Next, a quantitative comparison of the methods results in the best retrieval method in this experiment. All methods

[1] The RISM database can be queried on: http://www.rism.info/

[2] As an alternative to the RISM search engine, Monochord can be queried on: https://www.projects.science.uu.nl/monochord/risma2/

are used in determining which method is best suited for the purpose of deanonymising melodies using a k-NN. After a quantitative analysis, the best deanonymisation method is qualitatively evaluated by manually checking the plausibility of the composer labels given to *Anonymus* incipits.

2. METHODS

2.1 Pairwise alignment of melodies

Several methods have been researched for modelling melodic similarity. Some examples of these are n-gram methods [4] and geometric methods [5], each with their merits and disadvantages. Alignment of melodies has been implemented before by Kranenburg *et al.*[6]. Their method compares two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ by taking two symbols from each sequence. These symbols can either be aligned, or there is a gap between the two. Using a substitution score and a gap score, the total alignment score of the two sequences is calculated. This alignment score is to be minimised, as the two most aligned sequences have the least difference in notes and the smallest gaps between two elements of the sequences.

Before alignment, the melodies are transposed using a histogram approach where the pitch shift that maximises overlap in histogram bins is chosen [3]. The Monochord search engine employs these techniques to retrieve similar melodies [3]. All melodies are represented in the `base40` representation.

2.2 Querying with the RISM engine

RISM shows the graphical notation of the incipits to the user, but internally represents these incipits in the Plaine&Easie encoding, with strings such as `'4F8-FA''C/4F8.At3GA'''4C/'F8-F''D/4F` denoting an incipit [7]. A melody is found by first creating a corresponding FAST-index of the Plaine&Easie encoding, where the code is reduced to only pitch values. The engine can then search with or without transposition. In the latter case, only the created FAST-index is used in the search. In the case of transposition, all transpositions of the original search string are added to the FAST-index. This search index is then matched on the existing RISM database [7].

2.3 Querying with the Monochord engine

Potential matches between two melodies are tested for their similarity using a similarity score. This score denotes how well two melodies match, where a higher score is a better match between the melodies. The similarity score of two melodies is calculated during their alignment. The calculation of the scores is where the selected search method plays a role. A search method in the Monochord search engine consists of three types of raters that calculate a subscore by deciding how well the melodies match in the area the rater is specialised in. The sum of these subscores is the overall score that is used for the ranking of the results. The melodies with the highest similarity score will be placed at the top of the list.

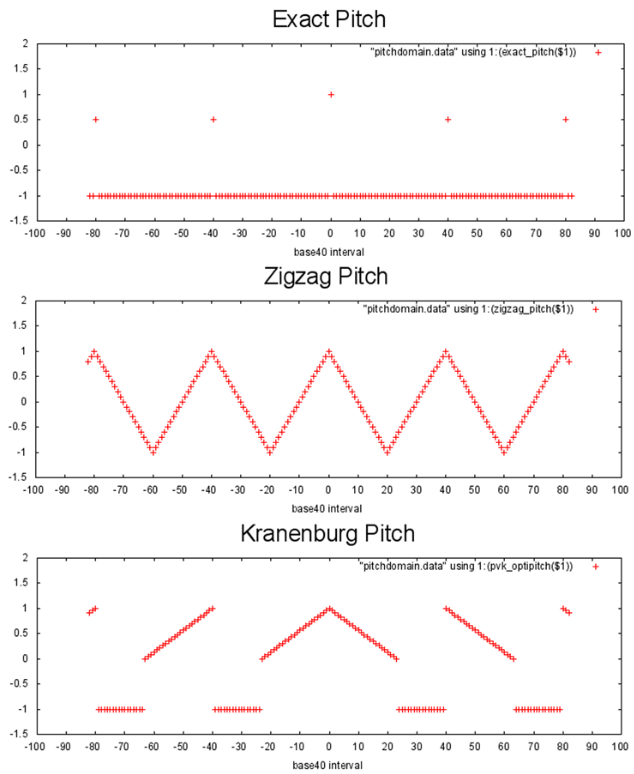


Figure 1. Graphs showing the score assigned to a difference in pitch in base40 representation for the exact pitch (pi2), zigzag pitch (pi3) and Kranenburg pitch (pi1) raters.

Monochord works with search methods that are a combination of three factors with three settings each. First, there is the category of pitch raters that return a value between -1.0 and 1.0. The settings are exact pitch (`pi2`), zigzag pitch (`pi3`) and Kranenburg pitch (`pi1`). The simplest one is exact pitch, which returns a score of 1.0 if aligned notes have an equal pitch, or a score of -1.0 if they differ. A difference of one or more octaves is assigned a score of 0.5. The zigzag pitch rewards notes that are close to each other and punishes notes that differ more. Notes of equal pitch return a score of 1.0. This score decreases linearly to -1.0 when the notes are most different at a distance of 20 in the `base40` representation and then increase to 1.0 where the notes differ by an octave at a distance of 40. The Kranenburg pitch is described by Peter van Kranenburg [6, 8]. For this rater, the score decreases linearly from 1.0 to 0 for intervals up to a fifth; large intervals up to the octaves receive a score of -1.0.

Graphs of the score assignment for each difference in pitch is shown for these raters in Figure 1 [3].

Secondly, there is the category of raters based on metric weight. The settings for this rater is no weight rater (`mw0`), ima weighted (`mw1`) and ima combined (`mw2`). The metric weight for these raters are computed with the inner metric analysis (ima) method by Volk [9]. With the ima weighted method, the influence of a note depends on its metric weight. The weights of the notes of each melody are scaled such that the average weight is 1. The value computed by the pitch rater is multiplied by the average of the weights of the two notes that are compared. The effect is that pitch difference on stressed notes have more influence than differences on less stressed notes. The ima combined method the metric weight has a more independent character than in the previous method. The absolute difference between the two metric weight values is considered and multiplied by the value produced by the pitch rater.

Thirdly, there is the category of duration-based raters. The settings are duration not included (`dur0`), fixed duration (`dur1`) and scaled duration (`dur2`). With no duration, the duration of the notes are not taken into account. With fixed duration, the difference in duration is taken as notated per incipit without any duration scaling. The scaled duration method uses histograms of the duration of notes. We use the duration scaling factor for the query melody that maximises the overlap of the histogram bins [3].

Each search method produces a result file that contains the first 50 ranked search results per query in the ground truth file. A search request consists of a search method and a query ID. Monochord aligns the query melody with all other melodies in the database and calculates the similarity score between the two melodies in a pair based on the search method. A higher score means more similarity to the query and thus a more relevant search result. The resulting melodies are ranked based on their similarity score. All of this is done automatically with a script. A perfect retrieval system will include all the result documents from the ground truth file as the highest ranked retrieved melodies [10]. In practice, we'll encounter melodies that are ranked lower, or melodies that do not appear in the result file at all. Misranking or missing a document affects the performance of a retrieval system.

2.4 Creating a ground truth

The 27 query methods are analysed with a ground truth set [11]. This set contains all relevant result melody RISM signatures per query signature. The retrieval results should show pairs that correspond with those in ground truth: this means that the search method is a good retrieval system. The 2005 MIREX evaluation set of Typke [2] is used for this purpose. Human experts on music were asked to find matches in the 2002 RISM database for a given melody. The participants didn't sift through the whole old database of half a million incipits. Some selective filtering excluded all but 300 incipits per query melody. This filtering was based on for instance large differences in pitch range, duration of the shortest versus the longest note, maximum interval between subsequent notes and editing distance between rhythm strings. This number of incipits was brought down to 50 by manually excluding the remaining incipits that were perceived as too different. Finally, the human experts ranked the 50 incipits based on their similarity to the query melody.

This ground truth data set contains 11 queries with about 10 resulting signatures per query. At the time of construction of Typke's data set, the RISM database contained about half a million melody incipits. The database that powers Monochord has doubled in size since the original ground truth research. It is reasonable to assume that some of the

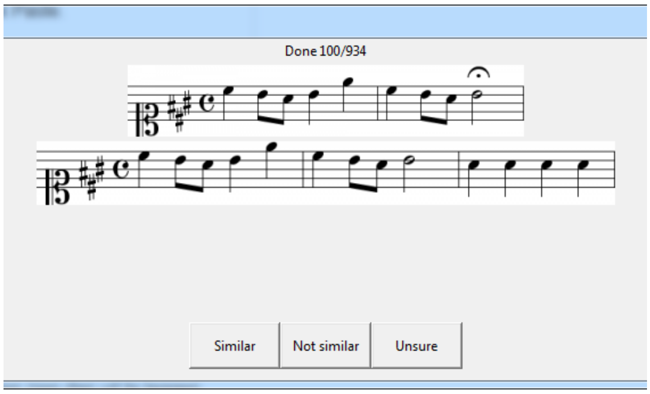


Figure 2. These two melodies are similar and the Similar button should be pressed in this case. Note how the query melody is the beginning of the result melody which has four additional notes.

additions may be truly relevant to one of the ground truth queries. Therefore, this ground truth set needs to be updated before a meaningful analysis of the query methods can be conducted. We update the set by manually checking all query-result pairs that appear in the ranked search results, but not in the original ground truth. These pairs are potential ground truth candidates, because the matching incipit could have been added after Typke's research was conducted. In total 6006 candidate items remain to be cross-checked for similarity by hand.

For this purpose, we create a comparison procedure for the query-result pairs. A computer program splits the candidates in batches of 1001 pairs and shows one pair at a time. The query is shown in musical notation on top, with the result below it. A human evaluator can press one of three buttons to confirm its comparison. The Similar button marks the pair as relevant (a line ending in a 1) and adds it to the ground truth file, then the pair is removed from the program's queue. Figure 2 shows a situation in which the Similar button must be pressed. The Not-similar button stores the pair as not relevant (a line ending in a 0) and removes the pair from the queue. The Unsure button is pressed whenever the evaluator can't make a decision at the moment, for whatever reason, and would like to go on with another pair. The pair is then added to the end of the queue and will return after all other pairs have been checked. The evaluator is shown a new pair after pressing one of these three buttons. Not all images of the musical notation are available on the RISM website, and thus they are unavailable in Monochord. Whenever such a pair comes by, it is handled as Not similar. All pairs without a definitive conclusion are handled as Not similar as well.

Completing one batch takes around 15-30 minutes and is less prone to learning effects that Typke described as possible shortcomings of the experiment [2]. Sequence effects might still occur, as all queries in sorted order. Filtering the pairs as Typke did is not necessary, because checking 6006 pairs manually is feasible. Yet some of the filtering techniques are subconsciously applied, such as rejecting absurdly long incipits or incipits with a greater pitch range instantly.

The ground truth has been expanded by adding 117 new relevant pairs. The new ground truth that is based on the original queries, but with additional results, is published and available for other researchers [3] .

2.5 Search method analysis

In evaluating the search methods, we are interested in the Mean Average Precision or area under curve (which are interchangeable terms). The search method with the best Mean Average Precision is designated as the best search method for this incipit database[12, 10, 13]. The methods are not only compared amongst each other, but also relative to an approximation of RISM's innate search engine. A reasonable approximation of RISM's retrieval method is using method `pi2mw0dur0`, as this uses only exact pitch in rating the melodies. This method can be seen as the baseline with which the other methods are compared.

Every search method is tested for precision and recall, which are plotted in the precision-recall curves. An important feature of our TREC files is the ranking of results. This ranking must be used in the evaluation of the search method. Several TREC evaluation tools have been made that utilise ranking (*trec_eval*[14], *trec_eval online*[15, 16], *pytrec_eval*[17]). We use the Python library *pytrec_eval* because the previously written scripts can be transferred to this task.

This evaluation tool requires two types of input: one file containing the expected results (the ground truth) and one file containing the retrieved results. The ground truth file and the result files are stored in the conventional TREC format. The TREC version of the ground truth file is filled with tab-separated lines that contain the RISM signature of the queried document, an iteration number Q_0, its result signature and a relevance rating [18]. Each line has the following format:

$$s_{query} \quad Q_0 \quad s_{result} \quad relevance$$

with $s_{query} \in RISMSignatures$, $Q_0 = 0$, $s_{result} \in RISMSignatures$ and $relevance \in \{0, 1\}$. The result file is similar to the ground truth file, but instead of a relevance rating, it returns a ranking for the document found. Additionally, each line contains a score, representing how well the result matches to the query, and a constant Exp. Both the score and Exp are ignored in this experiment by setting them to zero. A line in the result file has the following format:

$$s_{query} \quad Q_0 \quad s_{result} \quad rank \quad score \quad Exp$$

with $s_{query} \in RISMSignatures$, $Q_0 = 0$, $s_{result} \in RISMSignatures$, $rank \in \mathbb{N}$, $score = 0$, $Exp = 0$.

The evaluation tool uses TREC files, thus we need to convert the information stored in Typke's HTML files to this format. The ranking is not taken into account, all incipits ranked as relevant are used as is. Every melody is referred to with its RISM signature, which is precisely the format needed for our TREC files. For every incipit perceived as relevant with signature s_{result} in a file for a query with signature s_{query}, we create a line

$$s_{query} \quad 0 \quad s_{result} \quad 1$$

where the 1 at the end signifies this pair of query and result is a relevant pair, or a match.

2.6 Deanonymisation of melodies

Once the best-performing search method for retrieval based on melodic similarity has been determined, we can use this method to create data for the deanonymisation classification algorithm. We use a k-Nearest Neighbours algorithm to classify the anonymous melodies. A k-NN retrieves the label for the k elements that are most similar to the element that is to be classified. The most occurring label is said to be the classification of the unknown element. In the case of deanonymisation of melodies, the labels are composer names. As the search results provided by Monochord are ranked from most similar to least similar, we can simply take the top k results as the neighbours and use their metadata to get their composers.

The composer names are available in the RISM database as metadata of the melodies. This forms a mapping between all RISM signatures and their composers or an *Anonymus* label. The correct classifications are thus easily generated: it consists of looking up the melody signature in the mapping and then returning the composer-part in the metadata. If none of the neighbours have a composer label, the classification of the melody will simply be *Anonymus*.

Training and test data for the classification algorithm is widely available. Of the 1.2 million melodies, about a million have a known composer. We randomly sample an amount of incipits with known composer and split the sample 50%-50% in a training and test set.

During the training phase, we use cross-validation to get the best value for k. Here, we use a smaller set of 40 incipits, which is split in a training and test set. The cross-validation consists of testing a k-NN with a certain k on the provided training data. We perform hyperparameter optimisation for k by using a grid search to test all values $k \in \{1, 2, 5, 10, 20, 50, 100\}$ and all $n = 27$ search methods [19, 12]. This takes $\mathcal{O}(k \times n)$ trials and the computation is quite costly, thus we would like to minimise the amount of trials. We first test the k-values only on the best retrieval method and find a good value for k. Then, we use this k to trial all the search methods. Only $\mathcal{O}(k + n)$ trials have to be completed in this manner. The performance of all such k-NNs are compared, after which the combination of k and search method of the best k-NN is selected.

These best k-NN settings are used to initialise the final classifier. A full set of 100 incipits, split in a training and test set, is used for this phase of the experiment. The performance of the classifier is determined using the test data set. It is important to test on a set different than the training set, as overfitting could occur. Overfitting is visible whenever there is great performance on the training set, but poor performance on the new test data. The classifier is stable whenever the performance of the training and test sets is similar.

[3] The revised ground truth is available here: http://www.projects.science.uu.nl/music/resources/

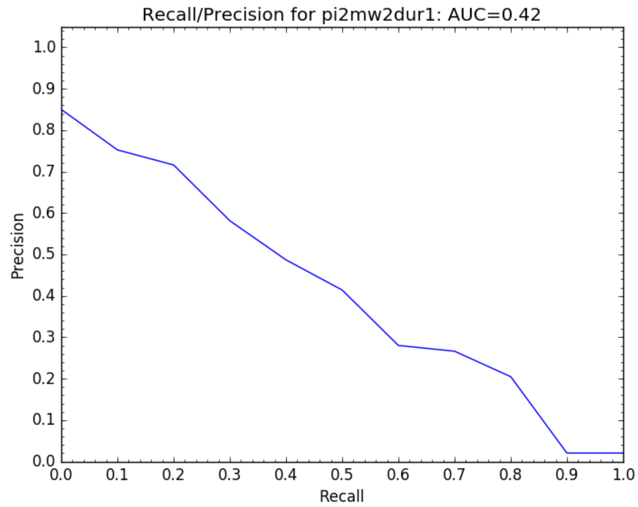


Figure 3. The precision-recall curve for retrieval method pi2mw2dur1. The area under curve, or mean average precision is 0.42, the highest in the series.

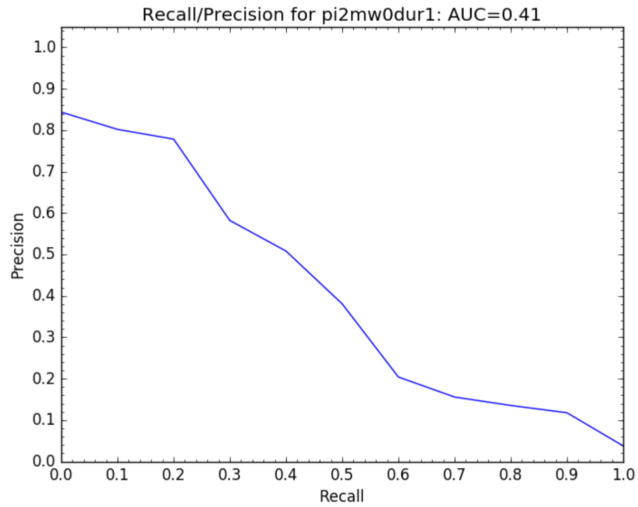


Figure 4. The precision-recall curve for retrieval method pi2mw0dur1. The area under curve, or mean average precision is 0.41, the runner-up in the series.

This trained classifier can in principle now be used to determine the composer of an anonymous song. This could be done for the 200,000 occurrences, but we randomly sample 100 melodies and evaluate some of the generated labels manually.

3. RESULTS

3.1 Search method analysis

Each of the 27 search methods produces a precision-recall curve from which the mean average precision is calculated. The mean average precision ranges from $0.03-0.42$ in the plots. The best method seems to be `pi2mw2dur1` (exact pitch, ima combined, fixed duration) with an area under curve of 0.42. The results are plotted in Figure 3.

Using exact pitch (`pi2`) gives the best results, with an average AUC of 0.38 in a range of $0.31 - 0.42$. The Kranenburg pitch (`pi1`) is the worst performer with an average AUC of 0.21 in a range of $0.03-0.32$. The exact pitch curves are plotted in Figure 8(b) in Appendix A, and the Kranenburg pitch as a comparison is shown in Figure 8(a) in Appendix A.

The best duration to use is fixed duration (`dur1`) with an average AUC of 0.33 (see Figure 10(b) in Appendix A). The best use of weight-based raters is by using none (`mw0`) with an average AUC of 0.35 (see Figure 9(a) in Appendix A).

These findings correspond with the best overall method, except for the weight-based rater factor. After a closer look, the method `pi2mw0dur1` seems to be a close runner-up with an AUC of 0.41 (see Figure 4). The overall performance of the ima combined (`mw2`) methods is not that different from `mw0` either, with an average AUC of 0.33.

The baseline approximation of the RISM search engine by using `pi2mw0dur0` results in an AUC of 0.35. Many of the search methods produced worse results than the baseline, but most of the exact pitch family produced equal or better results.

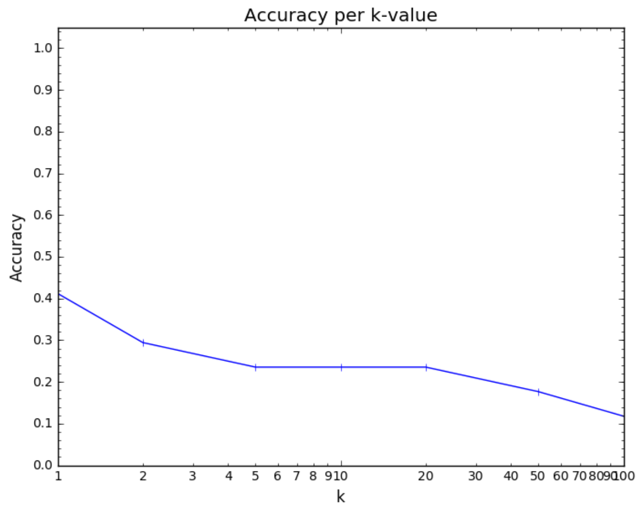


Figure 5. The accuracy of a k-NN for different values of k. The accuracy decreases with an increase in k.

A full table of AUCs for all search methods is specified in Appendix B.

3.2 Deanonymisation of melodies

Using 40 melodies as training set, we test the seven different values for $k \in \{1, 2, 5, 10, 20, 50, 100\}$ with the best retrieval method `pi2mw2dur1` to gain insight in the effect of the k-value on composer classification accuracy. Removing the *Anonymus* songs based on their ID is fallible process, as the existing set of IDs of known *Anonymus* songs turned out to be incomplete. There are still a few melodies with the *Anonymus* label hidden in the known data set. After filtering these out of the 40 melodies, we are left with 38 incipits.

The accuracy curve in Figure 5 shows that the accuracy decreases as k increases. This seems to have an intuitive reason, as with an increasing k, the share of wrong neighbours also increases. As the most similar songs are placed on the top, a low k will more likely consist of melodies with the wanted composer. The algorithm with a higher k will desperately try to come up with matches at the bottom, even when all the matching pairs have already been found. These bottom suggestions are more likely to be uninteresting, or even counterproductive, for composer classification. And yet the voting power of all these incipits is equal in a k-NN. If some composer turns up in the bottom results often enough, it will overthrow the correct decision made by the top results.

The best retrieval method might not be the best method for composer classification. Therefore another run is performed using $k = 1$ for all 27 search methods.

Method	Accuracy	Method	Accuracy	Method	Accuracy
pi1mw0dur0	0.211	pi2mw0dur0	0.263	pi3mw0dur0	0.263
pi1mw0dur1	0.158	pi2mw0dur1	0.158	pi3mw0dur1	0.158
pi1mw0dur2	0.211	pi2mw0dur2	0.211	pi3mw0dur2	0.158
pi1mw1dur0	0.053	pi2mw1dur0	0.263	pi3mw1dur0	0.158
pi1mw1dur1	0.000	pi2mw1dur1	0.263	pi3mw1dur1	0.053
pi1mw1dur2	0.000	pi2mw1dur2	**0.316**	pi3mw1dur2	0.106
pi1mw2dur0	0.211	pi2mw2dur0	**0.316**	pi3mw2dur0	0.263
pi1mw2dur1	0.158	pi2mw2dur1	0.263	pi3mw2dur1	0.211
pi1mw2dur2	0.158	pi2mw2dur2	**0.316**	pi3mw2dur2	**0.316**

Table 1. Table of accuracies per method, trained on the smaller set of 40 items. Bold numbers signify the highest accuracy.

The methods with the highest accuracy are `pi2mw1dur2`, `pi2mw2dur0`, `pi2mw2dur2` and `pi3mw3dur2` (see Table 1). The best retrieval method `pi2mw2dur1` has the second-highest accuracy, which will therefore also be considered in the possible parameters.

The k-NN is now trained on values for $k \in \{1, 2, 5\}$ and on the methods `pi2mw2dur1`, `pi2mw1dur2`, `pi2mw2dur0`, `pi2mw2dur2` and `pi3mw3dur2`. The data consists of 100 incipits randomly selected from the known melodies. These items are split in a 50% training set and a 50% test set.

The best classifier parameters turned out to be $k = 1$ with the `pi2mw2dur1` method (see Table 2). These settings resulted in a maximum accuracy of 0.375 on the test set.

	k=1	k=2	k=5
pi2mw1dur2	0.354	0.292	0.271
pi2mw2dur0	0.354	0.313	0.271
pi2mw2dur1	**0.375**	0.354	0.313
pi2mw2dur2	0.354	0.316	0.271
pi3mw2dur2	0.354	0.333	0.271

Table 2. Table of accuracies per parameter setting, trained on the full set of 100 items. The bold number signifies the highest accuracy.

Of the 100 melodies, 58 were given a non-*Anonymus* label. The guessed composers of the first eight such entries are given below in Table 3.

Using RISM's search engine [1], we find that incipit 450.202.307-1.1.1 classified as *Sperger, Johannes* indeed contains that name in the list of previous owners of the manuscript. The manuscript was put together by Joseph Michael Zink. This label seems plausible.

Signature	Composer
450.202.307-1.1.1	Sperger, Johannes
851.002.964-1.1.1	Werner, C.
702.020.071-1.1.1	Simonis, Ferdinando
240.006.107-1.1.1	Spohr, Louis
650.007.101-1.1.2	Meyerbeer, Giacomo
500.195.253-1.2.1	Paisiello, Giovanni
150.204.949-1.1.1	Gräfe, Johann Friedrich
454.013.591-1.1.1	Kluger, Johann Florian

Table 3. Table of deanonymised incipits and their composer labels in random order.

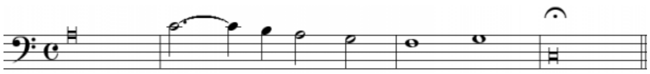


Figure 6. The incipit for 650.007.101-1.1.2, which is the original query (by *Anonymus*).

Incipit 851.002.964-1.1.1 classified as *Werner, C.* contains limited information besides the musical notation, thus a check is impossible.

Incipit 702.020.071-1.1.1 classified as *Simonis, Ferdinando* is called *Les noces?* and is part of a collection of French and Italian songs produced during Simonis' lifespan. The incipit is said to be arranged by the Frenchman André Jean Baptiste Bonaventure Dupont, and its manuscript is stored in Saint Omer's (France) public library. It seems more likely that Dupont is the composer of this song. The label of this incipit is questionable.

Incipit 240.006.107-1.1.1 classified as *Spohr, Louis* is called *Da wir uns niemals wieder finden in B-Dur* and is part of a collection of principally German melodies. The collection originated in 1808, which is during the German Spohr's lifespan. While this contextual evidence seems to make the attribution of this incipit to Spohr plausible, using any of the other search methods shows an abundance of related incipits by Mozart. Indeed, this incipit is a piece by Mozart. This example shows that contextual and music notational inspection are complementary methods of composer attribution analysis.

Incipit 650.007.101-1.1.2 classified as *Meyerbeer, Giacomo* is called *Falsibordoni* in Phrygian mode and is part of a homonymous collection of the same melody in different modes. This collection was put together in 1880, while Meyerbeer died in 1864. Still, it is possible that a piece Meyerbeer wrote was transposed in all modes after his death. Meyerbeer started studying music in Italy in 1816 [20], which explains the Italian name and the collection is stored in the Italian *Archivio diocesano*. The context seems to make the composer classification plausible, but once more music notational analysis shows otherwise. The query for 650.007.101-1.1.2 is shown in Figure 6, while the resulting Meyerbeer composition 452.020.643-1.2.1 is shown in Figure 7 for comparison.

Incipit 500.195.253-1.2.1 classified as *Paisiello, Giovanni* is called *Tenebre e pianto siamo in F-Dur* and is part of a

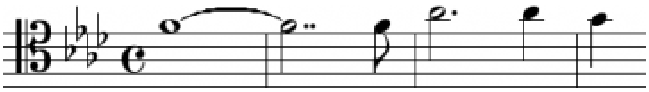


Figure 7. The incipit for 452.020.643-1.2.1, which is the resulting incipit (by Giacomo Meyerbeer). Notice how the incipit differs from the one in Figure 6.

collection of two other Italian melodies, produced in 1770. This is halfway through the Italian Paisiello's life, which makes this attribution plausible.

Incipit 150.204.949-1.1.1 classified as *Gräfe, Johann Friedrich* titled *Ich hab' es oft gesagt in G-Dur* is part of a collection that was produced during Gräfe's life, and his name appears next to one other melody in this collection: *Getrost mein Sinn erheitre dich in F-Dur*. Furthermore, the University of California owns another collection [21] that includes melodies of Gräfe, along with one called *Ich hatt' es oft gesagt in B-Dur*. The RISM ID of this collection is 000.114.155, which indeed gives us the melody with signature 000.114.246-1.1.1 that is an identical, but transposed, copy of our original incipit. This is a confirmed label.

Incipit 454.013.591-1.1.1 classified as *Kluger, Johann Florian* contains limited information besides being a part of a collection with dances exclusively written by Friedrich Joseph Kirmair and Josef Gellert. This collection contains solely German titles and storage locations, while all of the collections in the RISM database including Kluger's works are located in Czech libraries. The only information in favour of this label is the overlap in timespan of the three composer's lives and the 1800-1824 timestamp of the collection. The correctness of this label is questionable.

Three out of eight labels turned out to be quite plausible guesses. This precisely corresponds with the accuracy of the best k-NN classifier found using a test set. This finding makes the classifier results more convincing.

It took a fair amount of time to manually check these labels, but it takes significantly less time than having to come up with an initial guess via human effort. An effective strategy proved to consist of three stages. First, the collection the incipit is from can be scanned for similarities in composers or titles. Next, the timespans of the composer's lives and song publications should correspond. Another strategy is to compare languages, storage locations of the manuscripts, and country of birth or other important locations in the life of a composer. A final (or perhaps first) check is to analyse the matches by musical notation, as this will sometimes conflict with the results found in the contextual analysis. A good amount of music historical knowledge is necessary for this manual effort of label checking.

4. CONCLUSION

The search method `pi2mw2dur1` gives the best melodic similarity retrieval results. The method `pi2mw0dur1` is a good second choice, and might even be preferred when computation cost is factored in, as the ignored factor doesn't need to be calculated. Using exact pitch seems to be much more accurate than any other pitch rater, while the other settings do not matter as much and can be toggled off for an increase in speed.

Whether the best scoring methods are truly nearly equal in results is an interesting topic for further research. We assumed the RISM search engine uses a search method that is equal to `pi2mw0dur0`, it only uses exact pitch, and used that method as a baseline for the other search methods. To provide a true comparison with RISM's search engine, we would have to request the results for our ground truth queries and use this as the baseline. This is an opportunity to make our findings more reliable, yet under our assumption we expect that our claims will remain the same. Another point of improvement would be to look at the complementarity of search methods. Whereas the search methods are analysed in isolation, it might be possible that certain methods are suitable for one type of melody, while another method covers other types. Together, the range of accurate retrievals might be greater than they would be in isolated methods.

The deanonymisation process with a k-NN as described above has an acceptable accuracy (three out of eight plausible labels in the manual check), but the procedure is not accurate enough to become automated. A suggestion for further research would be to check the resulting labels more vigorously, and to do this for more classifications than the eight offered in this paper.

The accuracy of a stand-alone program for deanonymisation of incipits is questionable, but we've shown that using computerised suggestions from classification algorithms can help reduce the manual labour of labelling the songs. The most cost-efficient approach seems to be a combined effort of a computer scientist reducing the search space and offering composer suggestions to a music historian who analyses only a handful of possible composers, instead of the thousands the problem originally started with. For the purpose of giving suggestions, or narrowing the possible composers down to merely a few names, interesting follow-up research would be to test the accuracy of a k-NN that returns multiple labels. Instead of returning the best label, such a k-NN could return the top N composer suggestions. Whenever the true label is in this set of N labels, it is marked as correct. This will result in an equal or higher accuracy as the original k-NN used in this paper (a multiple label k-NN with $N = 1$), as the first result is always the same, with the multiple label k-NN having the benefit of having additional guesses. Such a classifier could conceivably achieve an accuracy that is worth automatising, whose result would be a set of possible composers that the music historian has to inspect for each incipit.

Points for further research include using the manual verifying strategies as features in machine learning applications, such as a k-NN. Perhaps using the collection an incipit is in, the title and composer's language, and the timespans to make a labelling decision can increase the accuracy of deanonymisation classifiers.

This research suggests that improving RISM's innate search engine is worthwhile, as the performance of alternative search techniques was found to be better than the baseline.

Computerised suggestions for composer labels are found to be a promising topic with room for improvement.

5. REFERENCES

[1] RISM, [accessed 2016-07-01]. [Online]. Available: http://www.rism.info/

[2] R. Typke, M. den Hoed, J. de Nooijer, F. Wiering, and R. C. Veltkamp, "A ground truth for half a million musical incipits," *Journal of Digital Information Management*, vol. 3, pp. 34–38, 2005.

[3] Monochord, [accessed 2016-07-01]. [Online]. Available: http://www.projects.science.uu.nl/monochord/

[4] J. Wolkowiczand and V. Keselj, "Analysis of important factors for measuring similarity of symbolic music using n-gram-based, bag-of-words approach," *Advances in Artificial Intelligence Lecture Notes in Computer Science*, pp. 230–241, 2012.

[5] G. Toussaint, "The geometry of musical rhythm," *Discrete and Computational Geometry Lecture Notes in Computer Science*, pp. 198–212, 2005.

[6] P. van Kranenburg, A. Volk, F. Wiering, and R. C. Veltkamp, "Musical models for folk-song melody alignment," *Proceedings of the International Conference on Music Information Retrieval*, pp. 507–512, 2009.

[7] J. Diet and M. Gerritsen, "Encoding, searching, and displaying of music incipits in the rism-opac," *Music Encoding Conference Proceedings, 2013 and 2014*, pp. 11–14, 2015.

[8] P. van Kranenburg, "A computational approach to content-based retrieval of folk song melodies," Ph.D. dissertation, Utrecht University, 2010.

[9] A. Volk, "The study of syncopation using inner metric analysis: Linking theoretical and experimental analysis of metre in music," *Journal of New Music Research*, vol. 37, no. 4, pp. 259–273, 2008.

[10] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. NY, USA: Cambridge University Press New York, 2008.

[11] J. Urbano and M. Schedl, "Minimal test collections for low-cost evaluation of audio music similarity and retrieval systems," *International Journal of Multimedia Information Retrieval*, vol. 2, no. 1, pp. 59–70, 2013.

[12] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[13] M. B. Kelly, "Evaluation of melody similarity measures," Master's thesis, Queens University, 2012.

[14] Text Retrieval Conference, "trec_eval," 2009, [accessed 2016-07-01]. [Online]. Available: http://trec.nist.gov/trec_eval/index.html

[15] S. Chatzichristofis, K. Zagoris, and A. Arampatzis, "trec_eval online," 2011, [accessed 2016-07-01]. [Online]. Available: http://thetrecfiles.nonrelevant.net/

[16] ——, "The trec files: the (ground) truth is out there," *Proceedings of the 34rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011.

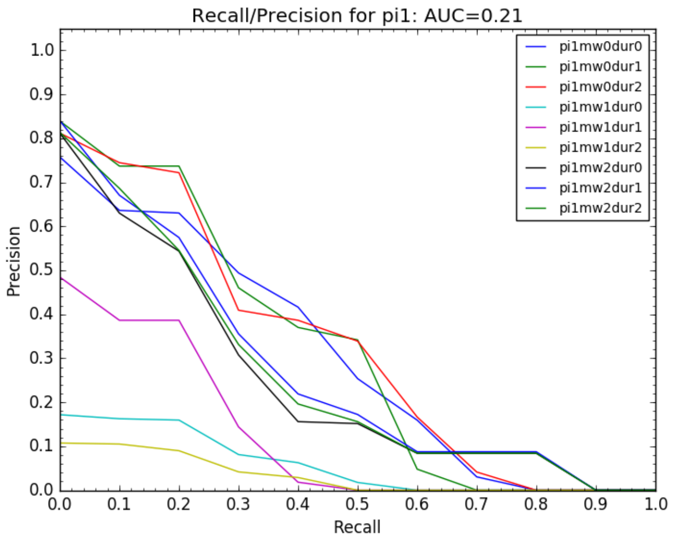
[17] "pytrec_eval," [accessed 2016-07-01]. [Online]. Available: http://www.github.com/XI-lab/pytrec_eval

[18] B. Peddi, H. Xiong, and N. ElSherbiny, "Trec_eval: Ir evaluation," 2010, [accessed 2016-07-01]. [Online]. Available: http://curric.dlib.vt.edu/modDev/package_modules/MidtermModuleTeam5-TRECevalFinal.pdf

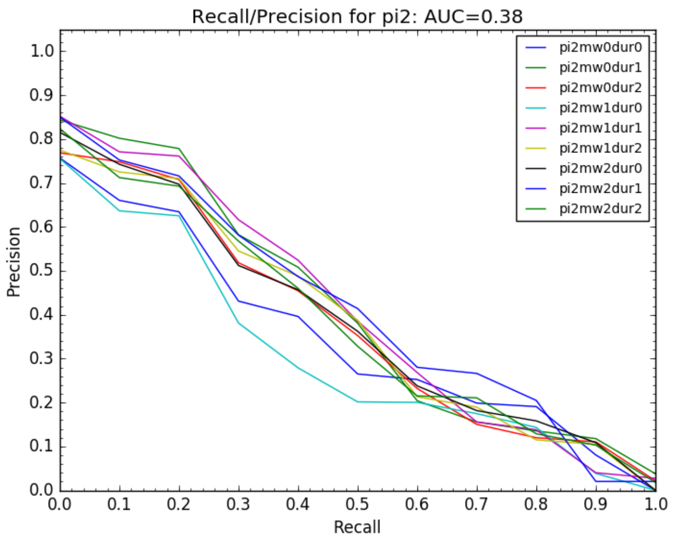
[19] A. B. Hassanat, M. A. Abbadi, G. A. Altarawneh, and A. A. Alhasanat, "Solving the problem of the k parameter in the knn classifier using an ensemble learning approach," *International Journal of Computer Science and Information Security*, vol. 12, no. 8, pp. 33–39, 2014.

[20] H. Becker, *Giacomo Meyerbeer in Selbstzeugnissen und Bilddokumenten*. Reinbek: Rowohlt Verlag, 1980.

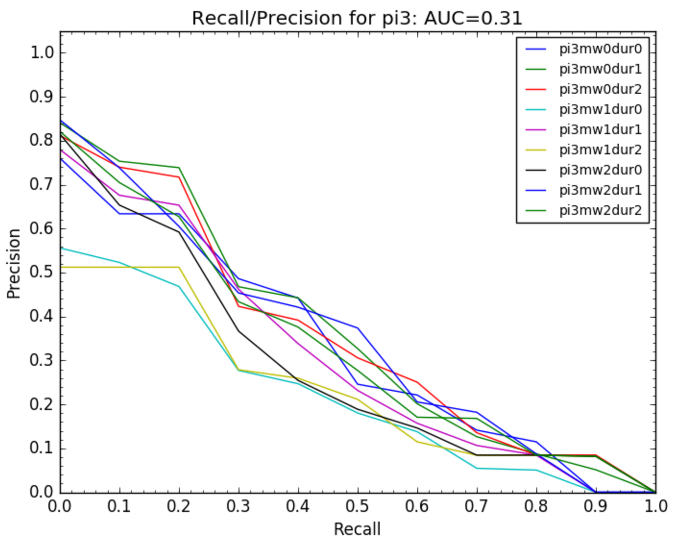
[21] "University of California collection," [accessed 2016-06-25]. [Online]. Available: http://beta.worldcat.org/archivegrid/collection/data/830325643



(a) All retrieval methods using Kranenburg pitch. The average AUC of the methods is 0.21.



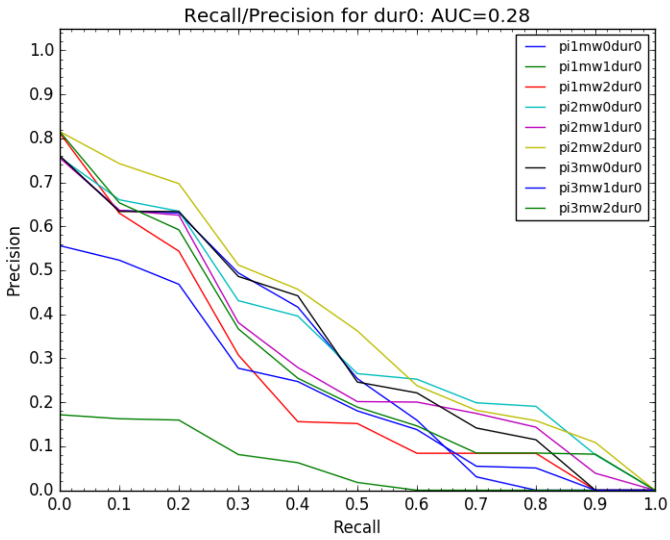
(b) All retrieval methods using exact pitch. The average AUC of the methods is 0.38.



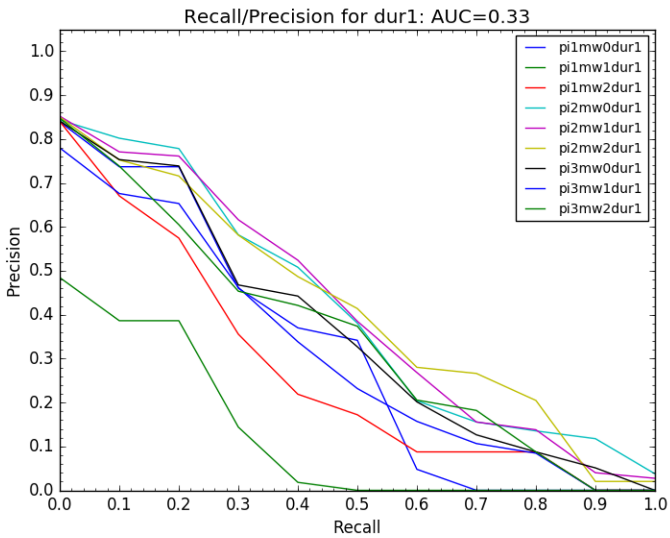
(c) All retrieval methods using zigzag pitch. The average AUC of the methods is 0.31.

Figure 8. The precision-recall curve for all methods from the pitch rater family. The average AUC of the methods is shown in the titles.

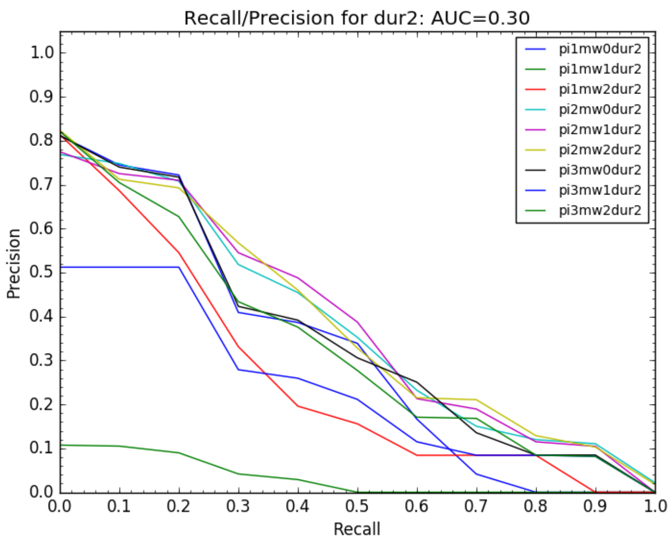
A. PRECISION-RECALL CURVES PER SETTING



(a) All retrieval methods using no duration rater. The average AUC of the methods is 0.28.

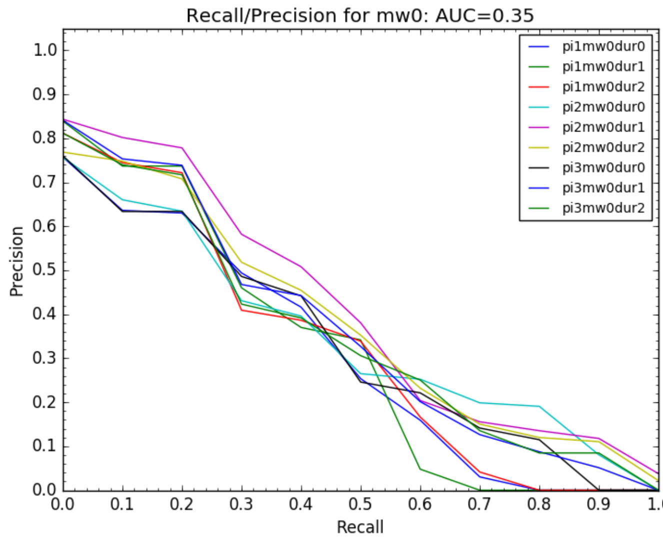


(b) All retrieval methods using the fixed duration rater. The average AUC of the methods is 0.33.

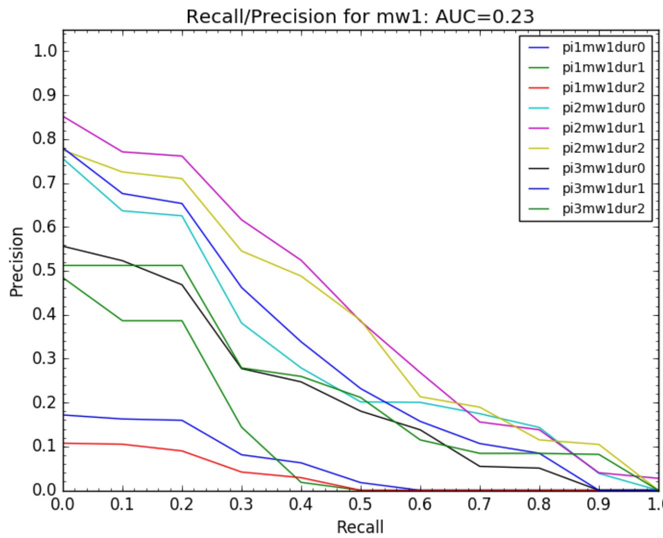


(c) All retrieval methods using the scaled duration rater. The average AUC of the methods is 0.30.

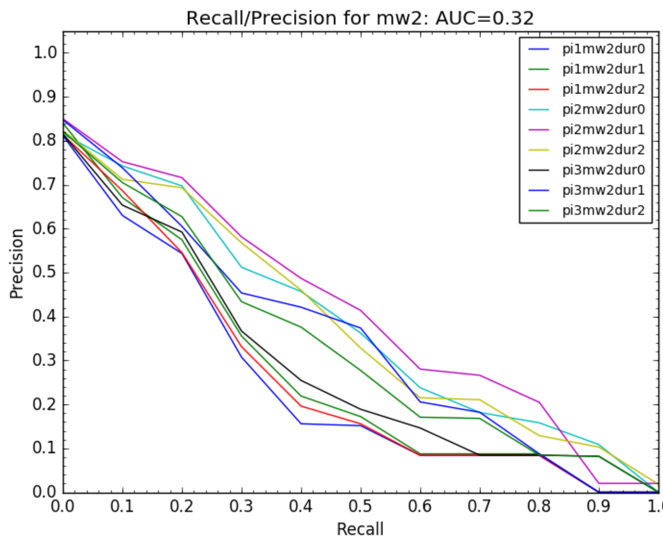
Figure 10. The precision-recall curve for all methods from the duration rater family. The average AUC of the methods is shown in the titles.



(a) All retrieval methods using no weight-based rater. The average AUC of the methods is 0.35.



(b) All retrieval methods using the ima weighted rater. The average AUC of the methods is 0.23.



(c) All retrieval methods using the ima combined rater. The average AUC of the methods is 0.32.

Figure 9. The precision-recall curve for all methods from the weight-based rater family. The average AUC of the methods is shown in the titles.

B. AUC TABLE FOR THE SEARCH METHODS

Method	AUC	Method	AUC	Method	AUC
pi1mw0dur0	0.30	pi2mw0dur0	0.35	pi3mw0dur0	0.33
pi1mw0dur1	0.31	pi2mw0dur1	0.41	pi3mw0dur1	0.36
pi1mw0dur2	0.32	pi2mw0dur2	0.38	pi3mw0dur2	0.35
pi1mw1dur0	0.06	pi2mw1dur0	0.31	pi3mw1dur0	0.22
pi1mw1dur1	0.12	pi2mw1dur1	0.41	pi3mw1dur1	0.31
pi1mw1dur2	0.03	pi2mw1dur2	0.39	pi3mw1dur2	0.24
pi1mw2dur0	0.24	pi2mw2dur0	0.39	pi3mw2dur0	0.29
pi1mw2dur1	0.27	pi2mw2dur1	0.42	pi3mw2dur1	0.35
pi1mw2dur2	0.26	pi2mw2dur2	0.38	pi3mw2dur2	0.33

Table 4. Table of the AUC for each of the search methods.