A Statistical Beat Induction Method for Metrical Grouping

Patrick de Kok, Gijs Kruitbosch, Nadya Peek
{pkok,gkruitbo,npeek}@science.uva.nl
Universiteit van Amsterdam

February 16, 2008

Abstract

When listening to music, humans can quickly find the beat to tap along to. In the field of Beat Induction, researchers attempt to simulate how the meter of a fragment of music is found. We have proposed a new computational model for finding the rhythm of a musical fragment. Our implementation uses a statistically oriented beat induction method, which harnesses a naive Bayes classifier to categorize beats and time signatures. Using a training corpus of folk songs, we match note placement and duration to different beats and time signatures using the WEKA machine learning toolkit. We evaluated several conditions for classification: to test how soon our system can generally correctly classify a beat; how soon our system can find a time signature; and finally to deal with possible difficulties with upbeats. When using our system to classify data, we find it performs quite well in distinguishing between ternary and binary time signatures, but does not do very well in classifying precise time signatures. In this paper, we offer our explanations for this and show the other results we have obtained using our classifier. We also discuss possible improvements and applications of our learning system.

1. Introduction

Humans quickly recognize beats and temporal patterns– they can generally find a beat to clap along with within a few seconds. Even with varying note-durations and tempo switches within a song, humans manage to find a regular interval to dance or clap along to. The way humans manage to deduce the isochronous pattern of the beat is however still poorly understood. Yet we manage, even without any musical training and from a very young age, to find the beat.

> *You're browsing, let us imagine, in a music shop, and come across a box of faded pianola rolls. One of them bears an illegible title, and you unroll the first foot or two, to see if you can recognize the work from the pattern of holes in the paper. Are there four beats in the bar, or only three? Does the piece begin on the tonic, or some other note? Eventually you decide that the only way of finding out is to buy the roll, take it home, and play it on the pianola. Within seconds your ears have told you what your eyes were quite unable to make out – that you are now the proud possessor of a piano arrangement of "Colonel Bogey"'*
>
> – H. Christopher Longuet-Higgins [8]

One of the ways beat is assumed to be induced is through selective listening. In many types of layered music, the beat is given by the lower tones- the bass drum or the lower octaves of a piano. However, it is also possible to deduce a beat from only a melody line. The beat then may either be deduced through the tonality of the piece (depending on the time signature and key of the piece, certain chords might be expected at certain times) or the duration of the notes (especially in pieces with a simple rhythmic structure). In this paper, we will attempt to induce beat with only information on note duration.

Note duration is directly coupled to time signature. Some of the questions we will ask during the course of this paper will include how easy it is to deduce the time signature given the beat, or how easy it is to deduce the beat given the time signature. It may for instance be easy to distinguish $\frac{3}{4}$ from $\frac{4}{4}$ but not $\frac{3}{4}$ from $\frac{6}{8}$.

Various computational models for beat induction have been proposed to simulate a beat induction process. The approaches taken for beat induction

vary widely: neural network approaches attempt to find beat through the constant reinforcement and decay of expected notes; rule-based systems which describe a set of rules encoding procedural aspects of knowledge; pattern recognition tasks which look to match a fragment to a previously heard beat; statistical approaches aiming to find statistical regularity in temporal patterns. We will provide a more extensive overview of these previous approaches in the next section 2.

The remainder of this paper is structured as follows: after the literature review section 2, we explain our approach to beat induction in section 3. We detail our implementation in section 4 and show our experiments and results in sections 5 and 6. Finally, we offer some conclusions and ideas for future work in sections 7 and 8.

2. Previous Work

This section is an outline of work done in beat and meter induction in the past.

2.1. Beat induction in humans

According to David Huron in *Sweet Anticipation* [4], listening to music evokes 5 response systems in humans: reaction; tension; prediction; imagination; and appraisal. The prediction responses reward the accurate prediction of the music because accurate prediction of events helps organisms better understand their environments. This is the reason why humans are so adept in predicting temporally structured sounds. The first two responses are also rooted in evolution, as they provide part of the fight-or-flight reaction mechanism. The last two have to do with human creativity, which is thus partially based on prediction.

2.2. Beat induction in AI

Music cognition has also been a very popular subject within the field of artificial intelligence, and many attempts to simulate various forms of music cognition have been made. Besides the problem of beat induction, in which one attempts to learn to tap along with the beat, attempts have also been made for meter induction, predicting consecutive chords and predicting melodic onsets. Below we will outline earlier research done in beat and meter induction, and explain how it relates to our approach.

2.2.1. Rule-based approach

Longuet-Higgins and Lee initially attempted to use formal analysis to find the meter of the anthems [7]. Here they try to make procedural aspects of knowledge more explicit by describing it through rules. Each rule contains a condition which recognizes a pattern in the input, as well as an action which then can modify the classification.

They initialize their learning algorithm by taking the duration between the first two onsets as the beat, and then incrementally adapt their prediction using a variety of rules like *stretch, update, conflate* and *confirm*. The rules are designed to revise the input and subsequently change the hypothetical meter, which is returned after a set time span. Longuet-Higgins and Lee assume here that an accented beat is more likely to fall on onsets higher in the tree, and that an accent in intervals may be perceived as a longer note duration, because longer note durations are generally more salient to the listeners.

The order in which the rules are applied and thus interact with each other is crucial. Longuet-Higgins and Lee do not propose a method for testing the manner in which rules interact with each other, which would be necessary should we want to expand their approach [2]. Furthermore, Longuet-Higgins and Lee begins as a rather simple rule based system, but as they attempt to cover more exceptions and possibilities they add more and more rules, inevitably leading to a more complicated system. The way these fine tuning rules are chosen by the authors seems somewhat arbitrary, which would not lead to a very robust classifier. If we were to continue Longuet-Higgins and Lee's approach, we would have to find a more systematic way to come up with new rules for the system.

2.2.2. Recurrent network approach

Another method to learn metrical grouping would be by means of (recurrent) neural nets. In this approach, a neural net is trained to develop expectancy for future events, based on the past events it has already observed. Such a method was attempted by Bharucha and Todd [1] to predict chord sequences in musical fragments. Useful attributes of neural nets are their ability to learn incrementally, without a complex control mechanism, and without a limit for input permissible for describing the context of the musical fragment.

However, there are some difficulties with a neural net approach. A general issue with learning by means of neural nets is that once trained, the

weighted functions are not easily understood by humans. Instead, they simply provide a black box which produces the correct output in the best case scenario, or does not produce the correct output with little to no explanation in the worst. Should we want our classifier to give us some insight into how beat induction works, then a neural net would not be an ideal choice.

An aspect of neural nets specific to our problem is that neural nets do not store temporal information spanned over a longer period of time very well, nor many different highly complex relations [11]. It is clear that these are two attributes we would very much like to see in our classifier, again making neural nets not a strong choice.

2.2.3. Reinforcing Oscillator Models

Large and Kolen attempted to make an oscillatory unit which can be networked into a system for entrainment and expectancy [5]. Each unit can track a different layer within a music fragment, thus allowing much more concurrent processing. Each unit is set in motion each time a note hits at the right moment, and is constantly adjusted as the music continues. With this approach they hope to overcome the two problems explained above, because the temporal aspect of music would be encoded in the expectancy of each oscillatory unit, and the complex relations can be realized through the mutual reinforcement of the units. Also, the system is more suitable for dealing with actual performed music, which will never have an entirely exact intervals.

2.2.4. Statistical approach

Another method to learn metrical grouping is to collect statistical regularities within a dataset of temporal data in the form of symbolic music with predefined meters. This could be done using a model which would predict the likelihood of different events at different positions in the bar, which could later predict which meters would be most likely given a symbolic music fragment. Palmer and Krumhansl [12] attempted this approach before using a collection of music scores to first determine the likelihood of certain meters given the occurrence of notes at a certain time within a measure.

2.2.5. Data-oriented processing

Van Zaanen, Bod and Honing have attempted to use Data-Oriented Processing to determine the possibilities of meters in symbolic musical fragments [14]. Here temporal data is described in the form of a tree-branch, and the classification is given by the fitting of a tree-branch into a slot in another tree. This approach however does not address the recurrent nature of music at all, as each tree branch is seen as an independent structure within the tree, and could adhere to an entirely different meter.

3. Approach

Most earlier approaches have their flaws when applied on our problem. For a very good working version of the approach of suggested by Longuet-Higgins and Lee [7], it might be a matter of adding only a few extra rules, but there is also the possibility that one needs to add a plethora of rules. Because the amount of rules which have to be added to their system is uncertain, we would like to use a way of automatically discovering new rules. As discussed in section 2.2.2, neural networks are not optimally suited to work on the type of information we want to experiment with. To build an implementation of the Large and Koolen approach [5] suited for our problem will consume a lot of time. As said in section 2.2.5, Data-Oriented Parsing does not address the recurrent nature of music. Because we want to develop a system which might work in analogy with a human listener, we do not find the method suggested by Van Zaanen, Bod and Honing suitable for our research. We do think it probable that a human listener has learned this knowledge in a statistical way, but maybe in another way than suggested by Palmer and Krumhansl [12].

We have decided to apply the naive Bayes classifier for the classification of the temporal structures within a musical fragment. This classifier is known to work surprisingly well in some domains when trained on a small set of attributes and very accurate when trained on a large training data set [10]. Another classifier we considered was the Bayes optimal classifier, which we however decided against using due to its high cost when applied to a large data set. [10].

3.1. Naive Bayes classifier

The naive Bayes classifier can be applied in situations where each individual instance can be described by a conjunction of its attribute values, and the target attribute has a finite domain. The classifier is trained in a supervised learning phase, where it is presented with training examples paired with their target attributes. Later it goes through a test setting, in which a set of attributes without classifications are fed in. It returns the class to which

it considers the instance to belong. This is done by finding the maximal probability of a hypothesis such as "This fragment has a meter of $\frac{3}{4}$" given the conjunction of the attributes.

This classifier searches for the most likely hypothesis h when presented with a n-tuple of attributes $\langle a_1, \ldots, a_n \rangle$ in a set of hypotheses H:

$$h = \underset{h_j \in H}{\operatorname{argmax}} P(h_j | a_1, \ldots, a_n)$$

When applying Bayes' theorem [10] to this expression, it can be rewritten as:

$$h = \underset{h_j \in H}{\operatorname{argmax}} \frac{P(a_1, \ldots, a_n | h_j) P(h_j)}{P(a_1, \ldots, a_n)}$$

As the denominator of the above expression is only for normalization, which adds no information when you are choosing the arg max, this can be left out. The term $P(h_j)$ is easily estimated by counting the occurances of each h in the training dataset H. $P(h_j$ is then estimated as $\frac{1}{|H|}$. However, this can not be used in combination with the probability $P(a_1, \ldots, a_n | h_j)$. It is likely that this only occurs once or does not occur at all. This is only a possible option when working with an enormous database in which many different instantiations of $P(a'_1, \ldots, a'_n | h'_j)$ are covered. Because of this, it is customary to take on an inductive assumption which states that all attributes $a_1, \ldots a_n$ are conditionally independent from each other given any h_j. With this assumption $P(a_1, \ldots, a_n | h_j)$ can be rewritten to be the product of the probability of each attribute a_i given h_j. This gives us the following expression which is much easier to compute:

$$h = \underset{h_j \in H}{\operatorname{argmax}} P(h_j) \prod_{i=1}^{n} P(a_i | h_j)$$

As one might expect, the inductive assumption does not hold for our case, as there surely is a conditional dependency between the attributes we want to pass to the classifier [10]. Nevertheless, the naive Bayesian classifier often remains a well performing learning system [16].

Other classifiers based on Bayes' theorem, like the Bayes optimal classifier, might outperform the naive Bayes classifier in some cases. As a trade-off, their computational complexity increases exponentionally with the number of training examples. This is a very detrimental property in our case: we use a fairly large training dataset, in an analogy with a human listener's musical experience.

3.2. Data representation

We are unaware of any tools which are able to transform the audible beat sequence into an appropriate representation for our classifier. We need to use a symbolic representation of music in order to be able to present the naive Bayes classifier with a conjunction of attributes. Because of this, we want to represent it in a different form than its natural, audible form. We use the same representation as used by Desain and Honing [2], which represents each note in a musical fragment with an integer value– indicating the length of that note with respect to the shortest note in that musical fragment. For each musical fragment, the representation in rhythmic units of the quarter note is given, so it is possible to translate the represented intervals back to their note lengths.

Hence, our data is presented in the form of lengths of intervals between note onsets. An example would be 3 2 2 3 2 2 which would represent a rhythm like `(|..|.|.|..|.|.)`. The data set represented in this way is the Essen Folksong Collection [13]. We used 5699 different entries of this collection. Please refer to table 1 for an overview of the composition of the used entries with respect to their binary and ternary character. For each experiment, we removed entries if their value for the target attribute occurred on less than 1% of the data set. The aim of doing this is to make sure that the classifier will have enough data for all possible values of the target attribute to try to reliably classify them. Because of this decision, we use a different number of musical fragments per training attribute, as displayed in table 2.

Table 1: The number of binary, ternary and deviant time signature entries selected from the Essen Folksong Collection.

Meter	Number of entries
2	3141
3	2555
5	2
7	1

3.3. Classification attributes

As said before, we want to find the relations between note sequences and the meter, beat duration and bar duration, which are the three basic properties of the temporal structures we are interested in. The meter of a piece of music, or its time signature, is a notation used to specify the beats per measure and the beat duration. The beat duration is the length of a quarter note in rhythmic units. The bar duration is the length of a bar, or measure, expressed in rhythmic units.

Table 2: The number of used training instances per attribute to train for.

Training attribute	Training instances	% of whole data set
Meter	5594	98.16
Beat	5641	98.98
Beat duration	5681	99.68
Binary/ternary	5696	99.94

To learn something, one needs to select the attributes to train the classifier on. We considered several attributes to train the classifier on:

As described in the beginning of this section, the *beat duration* is the length of a quarter note in beat measures. We suggest representing this as an integer value.

The *bar duration* also has a natural integer representation. There should be no such thing as a bar duration of 4.2 beats, and thus we can safely take an integer representation for this.

The *meter* is of a different nature because it is a nominal attribute. Because of this, our system is trained on a set of strings where each string is a representation of a meter in the training data set. An example would be, for a music fragment in $\frac{3}{4}$, the string value "3/4".

It would be a possible option to learn from the tempo of a piece of music. This has been suggested as a rhythmical device aiding metrical determination in musical cognition by Meyer [9]. The tempo is the number of beats per minute at which a piece of music should be played. Previous research by Meyer [9] as well as Desain and Honing [3] has suggested that the speed of the music may help disambiguate which meter the music is using (e.g. $\frac{6}{8}$ music is not generally played extremely fast). This attribute should be supplied to the classifier as a positive integer value.

We also are interested in training the system by rhythm prototype matching. Rhythm prototype matching matches the note intervals in a piece to a prototype of a meter. Such prototypes can be subdivided into levels, which are subdivisions of a measure. For example, a $\frac{3}{4}$ rhythm would first be divided into 3 different portions (with accents on all the $\frac{1}{4}$ beats), then each of those in 2 different subportions (with subtle accents on all the $\frac{1}{8}$ beats), and only then each portion would be divided in two again (for $\frac{1}{16}$ beats). The matches with the actual notes with respect to the comparing prototype are then summed per level of such a tree figure. This example is visualized in Figure 1. A $\frac{4}{4}$ rhythm would be divided in 2 (accent on the first and third $\frac{1}{4}$ beat), then in 2 again (accents on all $\frac{1}{4}$ beats) and in 2 again (accents on all $\frac{1}{8}$ beats). At level III where we are looking at a beat of length 1, we have 8 matches. There are 7 matches at level II, which looks at beats of length 2, and there are 6 matches at the top level, which looks at beats of length 4.

Using these rhythm prototypes we count the number of notes that start on each beat on a certain level of the prototype, and sum these per level per rhythm prototype. This way, we establish how well the beats of a piece's musical notes 'match' with different rhythms. This is then supplied to the classifier as integer values.

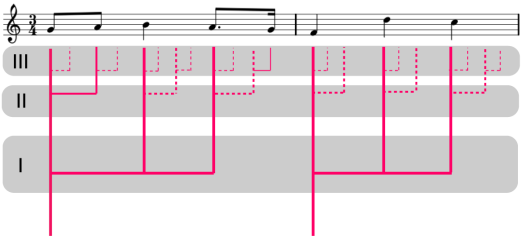


Figure 1: A small piece of music in $\frac{3}{4}$ matched against the $\frac{3}{4}$ rhythm prototype

It would be possible to just sum these attributes for each meter. However, knowing which and how many levels match is intended to aid the classifier in distinguishing subtle differences between similar candidate rhythms. For example, when trying to distinguish between $\frac{3}{4}$ and $\frac{6}{8}$, $\frac{3}{4}$ would be divided by 3, and then by 2 (matching every $\frac{1}{8}$ beat), whereas $\frac{6}{8}$ would be divided by 2, and then by 3 (also matching every $\frac{1}{8}$ beat). Clearly these will have the same value for the second level (all the $\frac{1}{8}$ meters) but different ones for the first level (all $\frac{1}{4}$ beats for $\frac{3}{4}$, the first and fourth $\frac{1}{8}$ beat for $\frac{6}{8}$). It is clear that specifying each level of each rhythm as a separate attribute here will be beneficial in distinguishing between the two rhythms. We could simply weight the matches of music fragments and prototypes at all different levels (by multiplying their values by a weighting constant) and sum over that,

hereby measuring one integer-valued attribute for each rhythm. However, after the summation there is no way to tell whether notes for instance only match one of the levels of the rhythm, so this approach would cause data loss.

3.4. Two different approaches

This method of rhythm prototype matching can be applied either top-down or bottom-up. When approaching the problem in a top-down fashion, the classifier is presented in the learning phase with musical fragments and a measure duration. It then matches each measure with the prototype for each meter, and classifies based on the results of these matches.

However, this approach also assumes that the system knows how long a measure is, as otherwise it would not be able to deduce how long all the different parts of levels of its meter prototype should be. This is rather counterintuitive, as one would normally assume that one does not yet know the measure length of a piece of music when trying to determine beat or a time signature.

Furthermore, this approach is not the approach a normal human listener applies. After listening to a short musical fragment, a human may already have found a model describing the temporal structures, and may be using it to tap along with the beat. The human will have also been able to make a judgment on the meter of the fragment before listening to it in its entirety.

The bottom-up approach instead matches the music with a monotonic beats, each having a different beat period. The number of matches can then be used by the classifier to decide upon which value of the target attribute best fits this piece of music. This does not require extra information, and could also be applied incrementally, making it more human-like.

3.5. Dealing with upbeats

We do foresee a certain problem in our approach. The classifier does not treat the upbeat of fragments differently from other parts of the fragment. The upbeat consists of a small number of notes before a full measure begins. The length of this sequence is always represented as a positive integer. Because the possible presence of an upbeat might bring the musical fragment out of phase with the beat, and because we intuitively think that there is no correspondence between upbeat length and the following note sequence, we expect this phenomenon to negatively influence the performance of the Naive Bayes classifier. We therefore would like to have several ways of coping with this problem in both the test and training phase of the classifier.

There are several alternative methods of handling upbeats. The most trivial attempt when the upbeat is known, is completely eliminating the upbeat. This can be done in two ways. The first is removing the note sequence which makes up the upbeat from the total musical fragment. Although you have less data in this way, all the data you have should contain only valid information of the temporal structure. The second method is putting a single note in front of the upbeat of sufficient length to make the upbeat a full measure long. This method preserves the data removed in our first suggestion, but it adds a "perfect" bit of data in there, and assumes that we already know the bar duration.

We also tested the default case where the upbeat is included without any alterations. In this case it is treated as any other part of the musical fragment. Although we do suspect this to have a significant negative effect on the outcome of the classifier, we sought to test this case.

Another possibility is using the *update* algorithm suggested in Longuet-Higgins and Lee [7], which tries to find the upbeat itself without any information except for the note sequence. It can be used by applying the algorithm of Longuet-Higgins and Lee until it has reached the point where it has recognized the end of the upbeat. It will return the position of the end of the upbeat and it would then continue with our own algorithm.

Our last suggestion to approach the upbeat problem is to shift the phase of the beat. Normally it is assumed that the prototype or monotic beat starts at the same time as the fragment does. But because there might be an upbeat in front of it, the two might be out of phase. We could try to find the right phase by gradually shifting the beat, one beat unit at a time, and matching the prototype or monotonic beat for every possible shift (between 0 and the length of the prototype or monotonic beat). This leads to different results than only removing the upbeats, or compensating them, as matching would be done exhaustively: if we are matching a monotonic beat of length n, there will be n different ways of applying this (shifting it by $0, 1, ..., n-1$), of which only one is 'right'. All the other shifts can give new information to the classifier, however, which might help it in making the right decision anyway.

4. Implementation

Although all approaches mentioned in the background review are valid and might work well, we have decided not to implement and test them all. The main reason for this is that we do not think it either belonging to the paradigm of learning systems, or not corresponding to the way a human listener might be doing a certain task. Instead, we implemented a statistically oriented method which measures how well a fragment of note durations matches with a prototype of a time signature.

4.1. Current Implementation

First, the corpus is preprocessed by a series of functions written in Common Lisp. It is at this stage that the upbeat and the length of the musical fragment might be altered, the fragment might be shifted, and the target attribute for the classifier is specified. Subsequently, we compare the notes with completely monotonic beats of increasing length. Whenever beats coincide, we increment a counter. This counter then represents the agreement between this particular regular beat and the actual notes. For example, comparing 2 2 1 1 2 1 1 with a monotic beat with beat length 2 will result in a score of 5, whereas comparing 2 1 1 3 1 1 3 would have a score of 4. We compare each piece of music to beats with beat periods up to 48, and use the scores obtained by these comparisons as attributes to train a Bayesian classifier to learn our target attributes meter, bar length, beat duration, and whether a piece's meter is binary or ternary.

In figures 2 and 3, we illustrate how such an approach could distinguish between $\frac{4}{4}$ and $\frac{3}{4}$, and compose the prototype of a meter that we used to illustrate this approach in section 3.3. The fragment (shown in pink dots) matches better with a regular beat of 3, which the classifier should be able to link to a $\frac{3}{4}$ time signature.

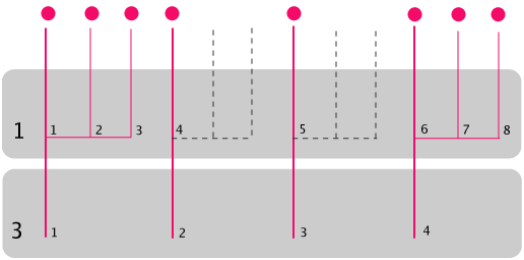


Figure 2: A fragment of note intervals matched with regular beats of 1 and 3 (corresponding to $\frac{3}{4}$, with no eighth notes). The score matching with a beat length of 3 is 4, the score of matching with a beat length of 1 is 8.

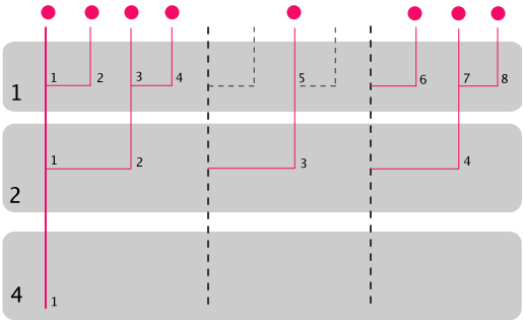


Figure 3: A fragment of note intervals matched with regular beats of 1, 2 and 4 (corresponding to $\frac{4}{4}$, with no eighth notes). Here the score of matching with a beat length of 4 is 1, for beat length 2 it is 4 and for beat length 1 it is 8 once more.

Finally we reformat the data set into the Attribute-Relation File Format (`.ARFF`), as used by Weka 3 [15], a data mining software suite developed at the University of Waikato. The `ARFF` files were subsequently loaded into Weka through its *Explorer* interface. The fragments have been classified with Weka's *NaiveBayes* classifier, using the matches with the different monotonic beats as attributes.

Our program works according to the bottom-up principle which we have described in section 3.4. We have varied over the different upbeat methods in the following way: each of the first three suggestions (*removal*, *compensation*, and *no-change*) have been applied to each note sequence. We have also added options to shift the matching as another way to add more data for the classifier and cope with upbeats.

Then we have the option of which target attribute we want to train the classifier for. The target attributes we want to train the system for are beat duration, bar duration, and measure length. As we do suspect it to be very hard to have the classifier discriminate between different binary or ternary beats such as $\frac{4}{4}$ and $\frac{2}{4}$ or $\frac{6}{8}$ and $\frac{3}{4}$, we have decided to also train for the target attribute of whether the fragment has a binary or ternary time signature.

We have trained for each permutation of above options an instantiation of the naive Bayes classifier, and we will analyze all the results in the next sections.

4.2. Unimplemented Approaches

As described in section 3, we have considered several options for our implementation. Although we would like to use tempo as one of the attributes to feed in, we have decided not to. The main reason for this is that we have not been able to find a musical corpus which has been digitized, and of a sufficient size, which has tempo annotated in it. Because of time limitations we were not able to expand an existing corpus with this information, or construct a complete new one.

The second approach we have decided not to implement is top-down prototype matching. Although this might work fine, it probably is not the way a human listener considers the stream of incoming notes when he or she is involved in beat identification. Even after a short period of time, the listener might already have formed a hypothesis of what the beat and beat length might be [6]. But this is not the case when a human tries to identify the meter of a piece of music. For doing this, a longer fragment is needed, and thus a top-down might still be applied.

Another reason for not implementing this is that we do not think this to be useful for all attributes we want to train on. And because we expect two attributes we want to train on, to determine the third, we do not find it necessary to implement this.

We tested all but one variations in upbeat handling. The *update* rule suggested by Longuet-Higgins and Lee [7] has not been implemented. This decision has been made mainly because of how it is embedded in their system of rules, and how dependent it is on those given rules. Because we presume the system should perform better when presented with more data, a method which is heavily dependent on fixed rules does not fit in our paradigm.

5. Experiments

When humans listen to music, they do not take in a whole song and then process it. They can already tap along with the beat after a couple of seconds [12]. Humans recognize the beat and participate. After training our learner, we hope it will be able classify accurately when presented with a short interval of note durations as well.

To model the way humans can incrementally deal with music fragments when they are presented, we compared how well our algorithm does with different amounts of information provided from the data. For this, we have separated our data into different categories. Do listeners determine meter or beat period after a certain amount of time? Or a certain amount of notes? How does analyzing part of a song compare to analyzing a whole song? Do humans achieve better estimates of bar duration or meter after being exposed to a larger fragment of music?

To deal with differing intervals, we have worked with both a maximum amount of total note duration as well as a maximum total number of notes. For comparison, we have also used an unlimited data set which runs the algorithm on the entire song. We expect the first two forms of sampling to provide similar results, and the unlimited experiment to outperform the first two with regards to accuracy but not with regards to complexity. It is however possible that after extensive training listening to the entire song becomes unnecessary to correctly classify a music fragment.

To reduce computational complexity, we have also made a distinction between using the algorithm on all intervals and only using the algorithm on intervals we deem significant. Significant intervals have a greatest common divisor with at least one of the numerators of the possible meters that is bigger than 1. More formally, an interval $i \in \mathbb{N}$ was considered significant in attempting to distinguish between meters in M when classifying songs if:

$$\exists x \in M \, \gcd(i, \text{numerator}(x)) > 1$$

holds, that is, if there exists at least one meter in M of which the numerator is not coprime with i. We have run the algorithm using first significant intervals, and then all intervals, to be able to make a comparison between the performance of the two. For example, if we would only want to distinguish between $\frac{3}{4}$ and $\frac{4}{4}$, we would not be interested in the interval 7. However, should we also want to be able to identify $\frac{7}{8}$, then 7 would become a significant interval.

These constraints result in 6 separate experiments of which an overview is given in table 3: looking at all intervals of unlimited data, looking at only the significant intervals of unlimited data, and then limiting the data on a certain amount of notes and a certain amount of total note duration, and measuring both all intervals and significant intervals on those.

Table 3: Overview of experiments

All intervals	Significant intervals
Unlimited data	Unlimited data
Limit amount of notes	Limit amount of notes
Limit total duration	Limit total duration

Additionally, we handled upbeats in three of the

different ways explained in section 3.5. First we removed upbeats entirely, then we added a long note at the beginning of the song to compensate the upbeat and form a full measure. Finally we also ran experiments in which we did nothing about the upbeat. We also added data for the same intervals shifted along the data, to help the learning algorithm recognize the right meter despite an upbeat.

To be able to do a true comparison, we would have to collect some data from human test subjects on how they do with beat classification. Due to time constraints, we have not done this, but instead will adopt the bias that humans are perfect classifiers of beat, and that for our dataset, it should be possible to achieve 100% accuracy.

6. Results

In this section we detail the results obtained with from the experiments that were explained in section 5. First we will explain the results obtained when classifying on all data available for each song, and then we will discuss the results from classifying on limited amounts of data. Finally we will detail some general findings related to the handling of upbeats

6.1. Classifying on all data

First we tested our approach with no limits on the sample size, so we classified based on the entire song. It turned out our approach did quite poorly in determining the exact meter of a piece. As can be seen in Figure 4(a), even in the best of cases it classified barely over 40% correct. Closer inspection of the result data (not reproduced) indicated that the confusion is primarily between meters that are very similar in note emphasis and beat patterns, such as $\frac{2}{2}$ and $\frac{4}{4}$. The results in figure 4(a) are from classifying using only data about significant intervals, but classification using all possible integer intervals between 2 and 48 did not improve the success rate of the algorithm.

Classification of beat (figure 4(b)) and bar duration (figure 4(c)) both produced relatively good results, averaging 60 to 70% correctly classified instances. This was somewhat surprising as it turned out several of the pieces in the data set had beat durations of less than 1 unit, which the algorithm didn't match against, as the smallest interval it used was 2 units (we didn't expect, at the time, that fractional units would ever be used by songs). This does explain the difference between the two, as bar duration detection does roughly 10 percentage points better than beat duration detection if upbeats are not removed or compensated for. This leads to a second important observation, which is that the presence of upbeats, even if there is shifted data available, greatly reduces the effectiveness of the algorithm when classifying bar durations.

Finally, figure 4(d) shows the results for detecting the 'type' of meter, that is, if it was binary ($\frac{2}{4}$, $\frac{4}{4}$, $\frac{2}{2}$, etc.) or ternary ($\frac{3}{4}$ or $\frac{6}{8}$). These results are quite encouraging, with the best strategies achieving as much as an 86% success rate. Here, the availability of shifted data does make a significant positive impact on the success of using the original data with upbeats present, unlike in many of the other cases.

All these results were produced using only the significant set of intervals, so those that were multiples of 2 and 3. This does not seem to have affected the results in a significant consistent way. The average absolute difference between two corresponding items with either only the significant or all the intervals was 2.27 percentage points, where half the approaches showed a better result for using all intervals, and the other half showing a better result for using only the significant intervals.

6.2. Classifying on limited data

After testing our approach on the complete data, we imposed limits on the amount of data given to the algorithm for processing, to simulate the human approach, which makes decisions before hearing the entire song. We used two different approaches to do this, one which limited the number of intervals (notes and rests) available, and the other limiting the total interval duration. We will outline the most significant trends here.

First of all, the differences between only significant or all the intervals were once again negligible. This further affirms that only using intervals which are multiples of the meters or beat durations that one is looking for is a sound approach to the various classifications that we were trying to make. This is important because it would significantly improve the speed of any on-the-fly processing that may need to be done in practical applications of the algorithm.

Second, surprisingly some of the classifications actually performed significantly better when given less data. This was especially true for the classification of bar durations when the total length of the sample was limited, and the upbeats were compensated. These results are displayed in figure 5(d). Here it is clear that the results for the compensated upbeats are between 5 and 10 percentage points higher than for the same classification given all the data of the song (visible in figure 4(c)). This

is probably caused by the fact that we compensate the upbeat by prefixing it with one long note in this algorithm, which means almost nothing will match inside this bar, until the first note of the upbeat. If the subsequent sample is sufficiently short, the bias will create the effect seen here.

Third, there is a very clear sudden increase in the success rate for the binary / ternary classification after given it more than 19 to 24 beat units of sample, depending on the method of upbeat correction used. This is visible in figure 5(a). We are not entirely sure what causes this sudden increase, apart from the obvious explanation that with less than 20 beat units, the algorithm does not have enough data to make a correct decision. However, this seems to contradict the fact that, for example, the bar duration classification works very well for limited amounts of data. Intuitively, one would assume that classifying whether something is a binary or a tertiary rhythm should be easier than classifying the bar duration, as typically the bar duration would be one of only a few multiples of the binary or tertiary rhythm, hence having a larger variance across the dataset.

Finally, when looking at the same classification with a limited number of intervals, a much more gradual change is observed (see figure 5(b)). We were not sure what caused this, or why this trend is so different from that for limited lengths (compare figure 5(a)). On average, adding one more interval increases the length of the sample by 2.2. So normally, a sample with 15 intervals is approximately the same as one of length 30, and one with 5 intervals should be approximately the same as one of length 15.

One possible explanation would be that cutting off at a certain length is less of a smooth transition: we may cut off much more in some cases (leaving a shorter sample than the limit), because leaving the next interval in would increase the sample length just a little bit too much. This would mean the increase in length along the edge of the graph would be less 'gradual', leading to the sudden increase in success rate as specific intervals (perhaps long intervals at the end of beats?) get included. We verified this on the data set, and it indeed seemed that, for example, if we do not remove upbeats, limiting samples to a length of 23 units leads to the average sample length being approximately 21.18, whereas for a maximum of 24 units, the average sample length is approximately 23.6. This, then, explains the sudden increase.

For completeness, we have included a table with the best results for the limits on the sample length as table 5, and a table with the best results for the limits on intervals as table 4.

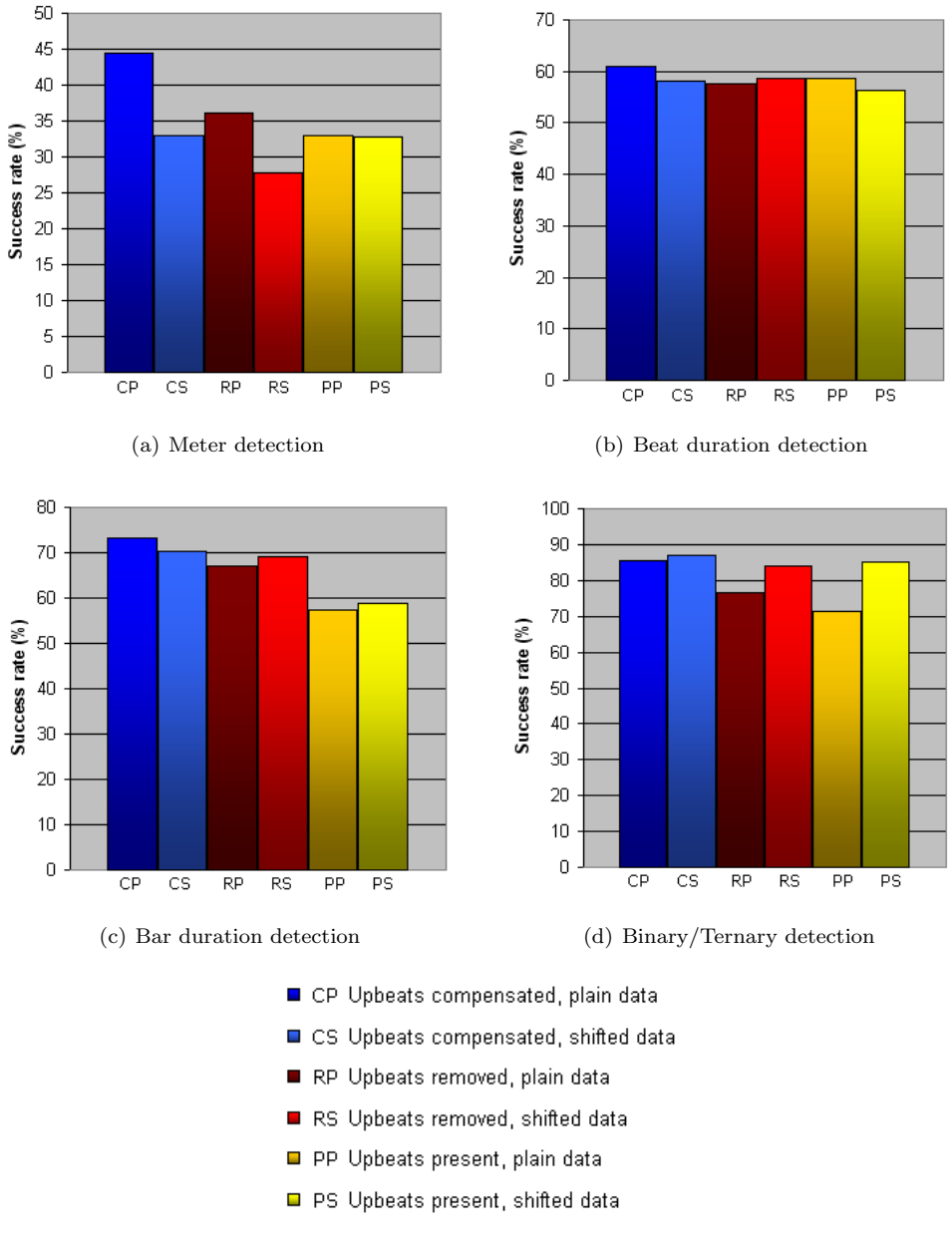
6.3. Upbeat handling

On the question of upbeats, there are several things that caught our eye after doing our experiments. First, It seems that providing shifted data does not help significantly, except in the case of classifying only if a song is using a binary or ternary type of meter (see figure 4(d)).

Furthermore, compensating upbeats with a long note tends to produce the best classifications, even if the amount of data is limited. This can be inferred from most of the figures, but is particularly clear in figure 4(a) and 5(d). This is intuitive in the sense that we are in fact presuming that we know how long the beat is, and how long the upbeat is, whereas for removing the upbeat, we only need to know how long the upbeat is, and for leaving everything as-is we do not need to know anything. These premises (the things we need to know) also influence what the result looks like, and it is therefore reasonable to expect that the final result is influenced positively by using as many of these premises as possible in modifying the original data set. In effect, we are 'cheating'.

One could argue that we should only have tested with upbeats present in all cases, but we were afraid this would not render useful results, and also thought that even if it did, it would be helpful to be able to compare them to results obtained without upbeats, or with compensated upbeats.

As it turns out, if one does not do anything to cope with upbeats, classification results do indeed suffer. If the amount of data is limited, even merely adding shifted data improves this. However, this is only enough to obtain reasonable performance if the amount of data is limited already, or if one is classifying whether the pieces of music are using binary or ternary meters. In all other cases, even with shifted data, not handling upbeats specially reduces the overall effectiveness of the classification by a sizable margin.



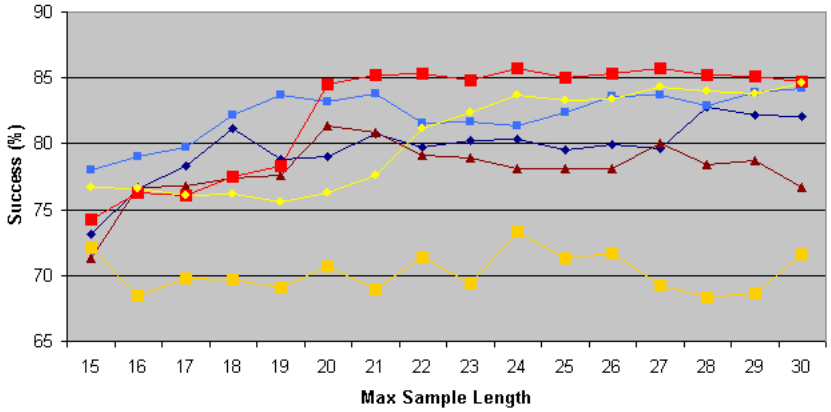
(a) Meter detection

(b) Beat duration detection

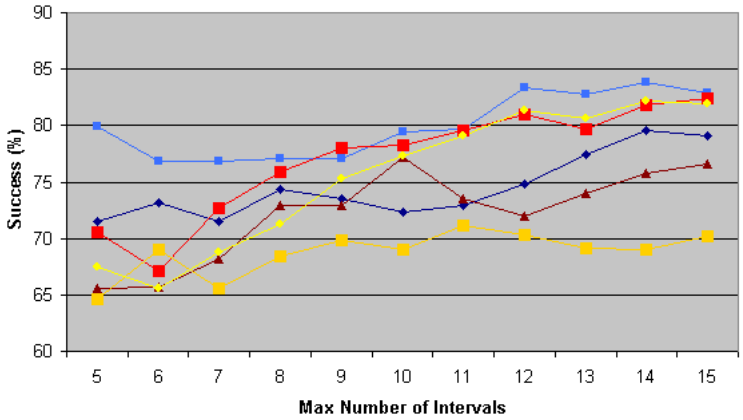
(c) Bar duration detection

(d) Binary/Ternary detection

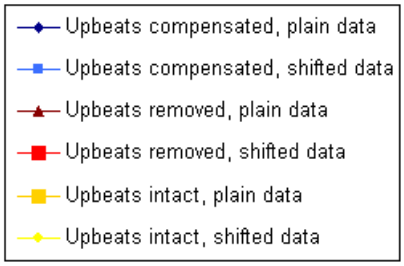
Figure 4: Detection of the different classes based on the significant intervals, with no limits on the number of intervals or the total duration, given different approaches to coping with upbeats in the music

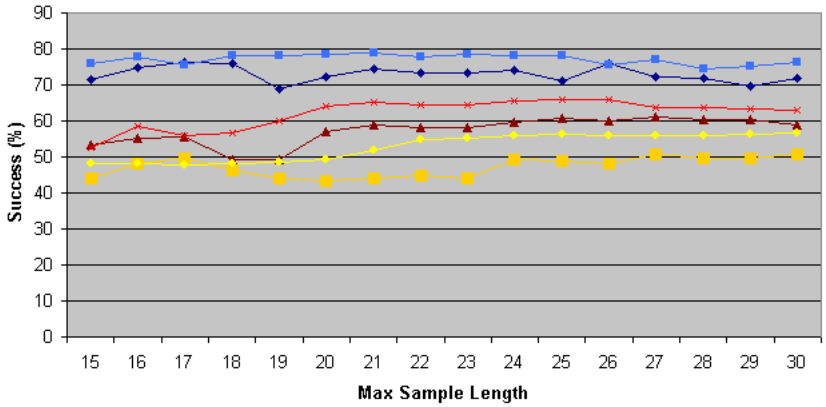


(a) Binary / Ternary classification with limited sample lengths and only significant intervals



(b) Binary / Ternary classification with a limited number of intervals and only significant intervals





(d) Bar duration classification with limited sample lengths and only significant intervals

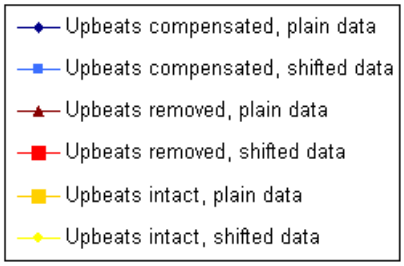


Figure 5: Selection of plots visualizing the effect of limiting sample size on classifying bar duration and binary vs. ternary meters.

Table 4: Success rate (in %) for the different classifications given a limited number of intervals, with only significant intervals used for classification

Number of intervals	Meter Signature	Beat Duration	Bar Duration	Binary / Ternary
5	36.81	63.67	72.88	79.92
6	30.68	61.40	71.05	76.81
7	30.32	56.91	66.46	76.86
8	33.63	59.55	68.14	77.04
9	35.38	58.60	67.75	78.00
10	34.39	57.74	69.49	79.46
11	35.48	58.04	69.62	79.71
12	35.48	56.56	71.28	83.34
13	39.11	57.63	70.34	82.72
14	39.65	58.07	69.77	83.84
15	40.88	64.28	69.88	83.92

Table 5: Success rate (in %) for the different classifications given a limited sample length, with only significant intervals used for classification

Number of intervals	Meter Signature	Beat Duration	Bar Duration	Binary / Ternary
15	36.31	64.20	76.01	77.98
16	37.79	64.34	77.81	79.04
17	37.88	63.74	76.16	79.78
18	38.70	64.81	78.04	82.18
19	37.61	63.58	78.02	83.74
20	39.47	63.86	78.66	84.48
21	39.52	63.42	78.80	85.20
22	38.65	62.89	77.77	85.32
23	38.81	62.68	78.62	84.87
24	39.31	63.12	77.96	85.78
25	38.22	62.96	78.16	85.01
26	39.15	62.89	75.89	85.31
27	40.31	63.07	77.08	85.78
28	40.81	60.68	74.61	85.25
29	41.26	62.52	75.02	85.13
30	40.36	63.32	76.21	84.69

7. Conclusion

Finding a beat in a piece of music is not as easy a task as it may seem at first glance. While using automated learning may seem a fine approach when considering downsides to static rules and other ways of tackling the problem, there are also significant issues. For example, what should be done about upbeats, and how should one filter and abstract the data so that it is easy for a classifier to learn patterns from it? We have tried to address these issues in our experiments. In this report we have explained several ways of working with upbeats, and have described how we implemented these. We have also reproduced the results of several experiments run with this framework and a database of folk songs.

With these experiments, we have been able to learn several important facts about using a Bayesian classifier to determine metrical information from musical intervals.

First, it is clear that classifying time signatures reliably with just these interval matches is too much to ask, given the low accuracy of the classifier for this target attribute. Whether a meter is binary or ternary, however, is relatively easy to determine.

Second, it is not generally useful to include intervals which do not correlate meaningfully with the meters you expect to need to classify when gathering data. The definition of significant intervals given in section 5 is useful here. It seems that intervals that are not related to the meters we want to distinguish between are only overhead.

Third, concerning the question of upbeats, there are several things that caught our eye after doing our experiments. It seems that providing shifted data does not help significantly in classifying correctly, except in the case of classifying if a song is using a binary or ternary type of meter (see figure 4(d)). Compensating upbeats with a long note seems to be the best overall method of classification, even if the amount of data is limited, and intuitively this does seem plausible. Leaving upbeats for what they are produces the worst classification results.

All these facts help being able to do practical things with a classifying system for beats. Although there are enough questions that remain unanswered (also see section 8 in this respect), we feel that the experiments we ran, and more importantly the results we obtained through them, are a good step towards productively being able to use Bayesian classification and interval matching to obtain a reliable indicator of beat.

8. Future Work

The described system works on symbolic input only, contrary to how our human listeners can perform. There is still a lot of research needed to develop a system which can interpret the note durations in real time. And when such a system has been developed, one could test it in combination with an adjusted version of our system.

But making a more 'human' system is not the only improvement which could be made. There is still enough room for improvement in automated classification of temporal structures such as meter, beat duration and bar duration. There is no system known which can perform in these tasks as well as a human listener. Although this system has more success than some earlier systems, there probably is a long route to a perfect system to analyze these structures.

Another point which still needs more research, is how well humans actually perform. We have assumed that it is possible for humans to learn to perform perfectly in the described classification tasks, but we were unable to find any references which show this.

References

[1] J. J. Bharucha and P. M. Todd. Modeling the perception of tonal structure with neural nets. *Computer Music Journal*, 13:44–53, 1989.

[2] P. Desain and H. Honing. Computational models of beat induction: The rule-based approach. *Journal of New Music Research*, 28:29–42, 1999.

[3] P. Desain and H. Honing. The formation of rhythmic categories and metric priming. *Perception*, 32:341–365, 2003.

[4] D. Huron. *Sweet Anticipation*. The MIT Press, Cambridge, MA, 2006.

[5] E. W. Large and J. F. Kolen. Resonance and the perception of musical meter. pages 65–96, 1999.

[6] J. London. *Hearing in Time: Psychological Aspects of Musical Meter*. Oxford University Press, 2004.

[7] C. Longuet-Higgins and C. Lee. The perception of musical rhythms. *Perception*, 11, 1982.

[8] H. C. Longuet-Higgins. The perception of music. *Proceedings of the Royal Society*, B 205:307–322, 1979.

[9] L. B. Meyer. *Emotion and Meaning in Music*. University of Chicago Press, 1956.

[10] T. M. Mitchell. *Machine Learning*. McGraw-Hill International Editions, 1997.

[11] M. C. Mozer. Neural net architectures for temporal sequence processing. *Predicting the future and Understanding the past*, 16:243–264, 1993.

[12] C. Palmer and C. L. Krumhansl. Mental representations of musical meter. *Joural of Experimental Psychology: Human Perception and Performance*, 16:728–741, 1990.

[13] H. Schaffrath. The Essen folksong collection. In D. Huron, editor, *Database containing 6,255 folksong transcriptions in the Kern format and a 34-page research guide*. Center for Computer Assisted Research in the Humanities, Menlo Park, CA, 1995.

[14] M. van Zaanen, R. Bod, and H. Honing. A memory-based approach to meter induction. In *Proceedings of the 5th Triennial ESCOM Conference; Hanover, Germany*, volume 6, pages 250–253, 2003.

[15] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.

[16] H. Zhang. The optimality of Naive Bayes. *Proceedings of the Seventeenth Florida Artificial Intelligence Research Society Conference*, pages 562–567, 2004a.