Journal of New Music Research
2004, Vol. 33, No. 4, pp. ••–••

Routledge
Taylor & Francis Group

Improved Methods for Statistical Modelling of Monophonic Music

Marcus T. Pearce* and Geraint A. Wiggins

Centre for Computational Creativity, City University, Northampton Square, London EC1V OHB, UK

Abstract

N-Gram based models have been used for a variety of musical tasks including computer-assisted composition, machine improvisation, music information retrieval, stylistic analysis and cognitive modelling. We present an application-independent evaluation of some recent techniques for improving the performance of a subclass of *n*-gram models on a range of monophonic music data. We have applied these techniques incrementally to eight melodic datasets using cross entropy computed by 10-fold cross-validation on each dataset as our performance metric. The results demonstrate that significant and consistent improvements in performance are afforded by several of the evaluated techniques. We discuss the results in terms of previous research carried out in the field of data compression and with natural language and music corpora and conclude by presenting some important directions for future research.

1 Introduction

Markov models have been applied to a number of musical research tasks including the development of practical applications and theoretical research. In the former category, we cite models for computer-assisted composition (Ames, 1989; Assayag et al., 1999; Hall & Smith, 1996), machine improvisation with human performers (Lartillot et al., 2001) and music information retrieval (Pickens et al., 2003) and in the latter, stylistic analysis of music (Conklin & Witten, 1995; Dubnov et al., 1998; Ponsford et al., 1999) and cognitive modelling of music perception (Ferrand et al., 2002; Reis, 1999a,b).

Our goal in the current paper is to investigate the performance of a range of such models on a variety of monophonic music data in an application independent manner. We are concerned, in particular, with the application to music data of a particular technique for combining the predictions of Markov models called *Prediction by Partial Match* (PPM – Cleary & Witten, 1984) which forms the central component in some of the best performing data compression algorithms currently available (Bunton, 1997). Outside the realm of data compression, PPM has been applied to natural language data (Chen & Goodman, 1999) and to music data (Conklin & Witten, 1995). Since its introduction, a great deal of research has focused on improving the compression performance of PPM models and our specific aim is to evaluate the performance of these improved models on a range of monophonic music. It is our hope that these improvements, evaluated here in an application independent manner, may then be applied usefully to some of the specific musical tasks cited above.

The paper is organised as follows. In §2, we give a general introduction to *n*-gram modelling as well as describing the PPM scheme in some detail. The information-theoretic performance metrics we shall use are also discussed. Much of the background for this research is drawn from the fields of statistical language modelling (Manning & Schütze, 1999) and text compression (Bell et al., 1990) since research in these fields is at a more mature stage of development than in the musical domain. However, we hope to demonstrate that practical techniques and methodologies from these fields can be usefully applied in the modelling of music. As noted above, *n*-gram models have been applied to a number of musical tasks and in §3, we discuss research in the musical domain which uses related models and methodologies. The data and experimental methodology employed are discussed in §4 where we also summarise the cross-product of PPM

Accepted: ■■

Correspondence: Marcus T. Pearce, Centre for Computational Creativity, City University, Northampton Square, London EC1V OHB, UK. E-mail: mt.pearce@city.ac.uk

DOI: ••

features to be evaluated. The results of our experiments are presented in §5 and discussed in §6. Finally, in §7, we conclude by presenting a number of useful directions for future research.

2 Background

2.1 Sequence prediction and *n*-gram models

For the purpose of describing this research we shall characterise the acquisition of knowledge about melodic music as a sequence learning problem (Dietterich & Michalski, 1986). The objects of interest are sequences of events where each event consists of a finite set of attributes and each attribute may assume a value drawn from some finite alphabet ξ. The simplified *musical surface* (Jackendoff, 1987) with which we shall be concerned consists of events corresponding to musical notes as notated on a score each of which consists of a single attribute corresponding to the chromatic pitch of the note. We shall use the notation $e_i^j \in \xi^*$ to denote a sequence of events $e_i \ldots e_j$ where $i \leq j \in \mathbb{N}^+$ and ξ^* denotes the set of all sequences composed of members of ξ including the empty sequence ε. The goal of sequence learning is to derive from example sequences a model which estimates the probability function $p(e_i|e_1^{i-1})$.

It is often assumed in statistical modelling that the probability of the next event depends only on the previous $n-1$ events, for some $n \in \mathbb{N}^+$:

$$p(e_i|e_1^{i-1}) \approx p(e_i|e_{(i-n)+1}^{i-1})$$

Under this assumption, we have an $(n-1)^{th}$ order Markov model or *n*-gram model. An *n*-gram is a sequence $e_{(i-n)+1}^{i}$ consisting of a *context* $e_{(i-n)+1}^{i-1}$ and a single-event *prediction* e_i. Since the use of a global order bound imposes assumptions about the nature of the data, the selection of an appropriate *n* is an issue when designing and building *n*-gram models. If the order is too high, the model will overfit the training data and fail to capture enough statistical regularity; low order models, on the other hand, suffer from being too general and failing to represent enough of the structure present in the data. The optimal order for an *n*-gram model depends on the nature of the data to which it is applied and, in the absence of specific a priori knowledge about that data, can only be determined empirically.

An *n*-gram *parameter* is the probability of the prediction occurring immediately after the context. The parameters are typically estimated on some corpus of example sequences. There are several different means of estimating *n*-gram parameters, the simplest of which is *maximum likelihood* (ML) estimation which estimates the parameters as:

$$p(e_i|e_{(i-n)+1}^{i-1}) = \frac{c(e_i|e_{(i-n)+1}^{i-1})}{\sum_{e \in \xi} c(e|e_{(i-n)+1}^{i-1})}$$

where $c(g)$ denotes the frequency count for *n*-gram g. In *n*-gram modelling, the probability of a sequence of events is expressed, following the chain rule, as the product of the estimated probabilities of the events (conditional on the identity of the previous $n-1$ events) from which it is composed:

$$p(e_1^j) = \prod_{i=1}^{j} p(e_i|e_{(i-n)+1}^{i-1})$$

When $n > i$, at the beginning of the sequence for example, padding symbols must be introduced to provide the necessary contexts.

Due to data sparseness, problems arise when using fixed order ML models due to the occurrence of as-yet-unseen *n*-grams. In particular, if a novel *n*-gram context is encountered or a symbol occurs in the data which has not previously appeared after an existing context (the zero-frequency problem – see Witten & Bell, 1991), the ML estimate will be zero. In these situations, the estimated probability of a novel *n*-gram will be too low and consequently the estimated probability of *n*-grams with non-zero counts will be too high. Additionally, as we shall see in §2.2, the information theoretic performance measures that we shall use require that every symbol is predicted with non-zero probability.

In statistical language modelling, a set of techniques known collectively as *smoothing* are commonly used to address these problems. The central idea of smoothing is to adjust the ML estimates in order to generate probabilities for as-yet-unencountered *n*-grams. This is typically achieved by combining the distributions generated by an *h*-gram model with some fixed *global order bound h* with distributions less sparsely estimated from lower order *n*-grams (where $n < h$). Most existing smoothing techniques can be expressed using the framework described in Equation (1) (Kneser & Ney, 1995).

$$p(e_i|e_{(i-n)+1}^{i-1}) = \begin{cases} \alpha(e_i|e_{(i-n)+1}^{i-1}) & \text{if } c(e_i|e_{(i-n)+1}^{i-1}) > 0 \\ \gamma(e_{(i-n)+1}^{i-1}) p(e_i|e_{(i-n)+2}^{i-1}) & \text{if } c(e_i|e_{(i-n)+1}^{i-1}) = 0 \end{cases} \quad (1)$$

For a given context $e_{(i-n)+1}^{i-1}$, if a given symbol e_i occurs with a non-zero count (i.e., $c(e_i|e_{(i-n)+1}^{i-1}) > 0$) then the estimate $\alpha(e_i|e_{(i-n)+1}^{i-1})$ is used; otherwise, we recursively *backoff* to a scaled version of the $(n-2)^{th}$ order distribution $p(e_i|e_{(i-n)+2}^{i-1})$ where the scaling factor $\gamma(e_i|e_{(i-n)+1}^{i-1})$ is chosen to ensure that the conditional probability distribution over the alphabet sums to one: $\Sigma_{e \in \xi} p(e|e_{(i-n)+1}^{i-1}) = 1$. The recursive step is typically terminated with the zeroth order model or by taking a uniform distribution over ξ. The various smoothing algorithms differ in terms of the techniques employed for computing $\alpha(e_i|e_{(i-n)+1}^{i-1})$ and $\gamma(e_i|e_{(i-n)+1}^{i-1})$.

An alternative to backoff smoothing is *interpolated* smoothing in which the probability of an *n*-gram is always estimated by recursively computing a weighted combination of the $(n-1)^{th}$ order distribution with the $(n-2)^{th}$ order distribution as described in Equation (2).

$$p(e_i|e_{(i-n)+1}^{i-1}) = \alpha(e_i|e_{(i-n)+1}^{i-1}) + \gamma(e_{(i-n)+1}^{i-1}) p(e_i|e_{(i-n)+2}^{i-1}) \quad (2)$$

Detailed empirical comparisons of the performance of different smoothing techniques have been conducted on natural language corpora (Chen & Goodman, 1999; Martin et al., 1999). One of the results of this work is the finding that, in

general, interpolated smoothing techniques outperform their backoff counterparts. Chen & Goodman (1999) found that this performance advantage is restricted, in large part, to *n*-grams with low counts and suggest that the improved performance of interpolated algorithms is due to the fact that low order distributions provide valuable frequency information about such *n*-grams.

2.2 Performance metrics

There exist many (more or less application dependent) ways of assessing the quality of an *n*-gram model and the ultimate evaluation metric can only be the impact it has on a specific application. Here, however, we are interested in examining the performance of such models in an application neutral manner. It is common in the field of statistical language modelling to use information theoretic, in particular entropy based, measures to evaluate statistical models of language. We have employed these metrics in this research and they are briefly introduced below.

Given a probability mass function $p(e \in \xi) = p(\chi = e)$ of a random variable χ distributed over a discrete alphabet $\xi = \{e_1, e_2, \ldots, e_k\}$ such that the individual probabilities are independent and sum to one, the entropy $H(p)$ is defined as:

$$H(p) = -\sum_{e \in \xi} p(e) \log_2 p(e) \tag{3}$$

Shannon's (1948) fundamental coding theorem states that entropy provides a lower bound on the average number of binary bits per symbol required to encode an outcome of the variable χ. The corresponding upper bound occurs in the case where each symbol in the alphabet has an equal probability of occurring: $\forall e \in \xi, p(e) = \frac{1}{|\xi|}$.

$$H_{max}(p) = \log_2 |\xi| \tag{4}$$

Entropy has an alternative interpretation in terms of the degree of uncertainty that is involved in selecting a symbol from an alphabet: greater entropy implies greater uncertainty.

In practice, we rarely know the true probability distribution of the stochastic process and use a model to approximate the probabilities in Equation (3). *Cross entropy* is a quantity which represents the divergence between the entropy calculated from these estimated probabilities and the source entropy. Given a model which assigns a probability of $p_m(e_1^j)$ to a sequence e_1^j of outcomes of χ, we can calculate the cross entropy $H_m(p_m, e_1^j)$ of model m with respect to event sequence e_1^j. In particular, if we make some assumptions about the stochastic process which generated the sequence, the cross entropy $H_m(p_m, e_1^j)$ may be calculated as:[1]

$$\begin{aligned} H_m(p_m, e_1^j) &= -\frac{1}{j} \log_2 p_m(e_1^j) \\ &= -\frac{1}{j} \sum_{i=1}^{j} \log_2 p_m(e_i | e_1^{i-1}) \end{aligned} \tag{5}$$

Cross entropy approaches the true entropy of the sequence as the length of the sequence (j) increases.

Since $H_m(p_m, e_1^j)$ provides an estimate of the number of binary bits required on average to encode a symbol in e_1^j in the most efficient manner and there exist techniques, such as arithmetic coding (Witten et al., 1987), which can produce near optimal codes, cross entropy provides a direct performance metric in the realm of data compression. However, cross entropy has a wider use in the evaluation of statistical models. Since it provides us with a measure of how uncertain a model is, on average, when predicting a given sequence of events, it can be used to compare the performance of different models on some corpus of data. In statistical language modelling, cross entropy measures are commonly used:

> For a number of natural language processing tasks, such as speech recognition, machine translation, handwriting recognition, stenotype transcription and spelling correction, language models for which the cross entropy is lower lead directly to better performance. (Brown et al., 1992, p. 39)

A related measure, *perplexity*, is also frequently used in statistical language modelling. The perplexity $PP_m(p_m, e_1^j)$ of model m on sequence e_1^j is defined as:

$$PP_m(p_m, e_1^j) = 2^{H_m(p_m, e_1^j)} \tag{6}$$

Perplexity provides a crude measure of average size of the set of symbols from which the next symbol is chosen – lower perplexities indicate better model performance.

2.3 The PPM algorithm

2.3.1 Overview

Prediction by PartialMatch (Cleary & Witten, 1984) is a data compression scheme the central component of which is an algorithm for performing backoff smoothing of *n*-gram distributions. Variants of the PPM scheme have set the standard in lossless data compression since its introduction (Bunton, 1997). We shall describe several of these variants in terms of Equations (1) and (2) where the recursive step is terminated with a model which returns a uniform distribution over ξ. This model is usually referred to as the *order* – 1 model and allows for the prediction of events which have yet to be encountered.

2.3.2 The zero-frequency problem and escaping

We shall now describe how the probability estimates $\alpha(e_i | e_{(i-n)+1}^{i-1})$ and $\gamma(e_i | e_{(i-n)+1}^{i-1})$ in Equations (1) and (2) are computed in PPM models. The problem is usually characterised by asking how we estimate $\gamma(e_i | e_{(i-n)+1}^{i-1})$ – the amount of prob-

[1] In particular, it is standard to assume that the process is *stationary* and *ergodic* (Manning & Schütze, 1999). A stochastic process is stationary if the probability distribution governing the emission of symbols is stationary over time (i.e., independent of the position in the sequence) and ergodic if sufficiently long sequences of events generated by it can be used to make inferences about its typical behaviour.

ability mass to assign to events which are novel in the current context $e_{(i-n)+1}^{i-1}$. $\alpha(e_i|e_{(i-n)+1}^{i-1})$ is then set such that the distributions sum to one. As noted by Witten and Bell (1991), there is no sound theoretical basis for choosing these *escape probabilities* in the absence of *a priori* knowledge about the data being modelled. As a result, although several schemes exist, their relative performance on any particular task can only be determined experimentally. In the following discussion, $t(e_i^j)$ denotes the total number of *symbol types*, members of ξ, that have occurred with non-zero frequency in context e_i^j; and $t_k(e_i^j)$ denotes the total number of symbol types that have occurred exactly k times in context e_i^j.

Method A (Cleary & Witten, 1984) assigns a frequency count of one to symbols that are novel in the current context $e_{(i-n)+1}^{i-1}$ and adjusts $\alpha(e_i|e_{(i-n)+1}^{i-1})$ accordingly:

$$\gamma(e_i|e_{(i-n)+1}^{i-1}) = \frac{1}{\sum_{e\in\xi} c(e_{(i-n)+1}^{i-1}) + 1}$$

$$\alpha(e_i|e_{(i-n)+1}^{i-1}) = \frac{c(e_i|e_{(i-n)+1}^{i-1})}{\sum_{e\in\xi} c(e|e_{(i-n)+1}^{i-1}) + 1}$$

As the number of occurrences of the context increases, $\gamma(e_i|e_{(i-n)+1}^{i-1})$ decreases and $\alpha(e_i|e_{(i-n)+1}^{i-1})$ approaches the ML estimate.

Method B (Cleary & Witten, 1984) classifies a symbol occurring in a given context as novel unless it has already occurred *twice* in that context. This is achieved by subtracting one from the symbol counts when calculating $\alpha(e_i|e_{(i-n)+1}^{i-1})$ and has the effect of filtering out anomalies. In addition, the appearance of the type count $t(e_{(i-n)+1}^{i-1})$ in the numerator of $\gamma(e_i|e_{(i-n)+1}^{i-1})$ has the effect that the escape probability increases as more types are observed.

$$\gamma(e_i|e_{(i-n)+1}^{i-1}) = \frac{t(e_{(i-n)+1}^{i-1})}{\sum_{e\in\xi} c(e_{(i-n)+1}^{i-1})}$$

$$\alpha(e_i|e_{(i-n)+1}^{i-1}) = \frac{c(e_i|e_{(i-n)+1}^{i-1}) - 1}{\sum_{e\in\xi} c(e_{(i-n)+1}^{i-1})}$$

Method C (Moffat, 1990) was designed to combine the more attractive elements of methods A and B. It is a modified version of method A in which the escape count increases as more types are observed (as in method B).

$$\gamma(e_i|e_{(i-n)+1}^{i-1}) = \frac{t(e_{(i-n)+1}^{i-1})}{\sum_{e\in\xi} c(e_{(i-n)+1}^{i-1}) + t(e_{(i-n)+1}^{i-1})}$$

$$\alpha(e_i|e_{(i-n)+1}^{i-1}) = \frac{c(e_i|e_{(i-n)+1}^{i-1})}{\sum_{e\in\xi} c(e|e_{(i-n)+1}^{i-1}) + t(e_{(i-n)+1}^{i-1})}$$

One particular smoothing technique called *Witten-Bell smoothing*, often used in statistical language modelling, is based on escape method C (Manning & Schütze, 1999).

Method D (Howard, 1993) modifies method B by subtracting 0.5 (instead of 1) from the symbol count $c(e_i|e_{(i-n)+1}^{i-1})$ in $\alpha(e_i|e_{(i-n)+1}^{i-1})$.

$$\gamma(e_i|e_{(i-n)+1}^{i-1}) = \frac{\frac{1}{2}t(e_{(i-n)+1}^{i-1})}{\sum_{e\in\xi} c(e_{(i-n)+1}^{i-1})}$$

$$\alpha(e_i|e_{(i-n)+1}^{i-1}) = \frac{c(e_i|e_{(i-n)+1}^{i-1}) - \frac{1}{2}}{\sum_{e\in\xi} c(e|e_{(i-n)+1}^{i-1})}$$

Method AX (Moffat et al., 1998) is motivated by the assumption that novel events occur according to a Poisson process model. On this basis, Witten and Bell (1991) have suggested method P which uses the following escape probability:

$$\gamma(e_i|e_{(i-n)+1}^{i-1}) = \frac{t_1(e_i|e_{(i-n)+1}^{i-1})}{\sum_{e\in\xi} c(e|e_{(i-n)+1}^{i-1})} - \frac{t_2(e_i|e_{(i-n)+1}^{i-1})}{\left(\sum_{e\in\xi} c(e|e_{(i-n)+1}^{i-1})\right)^2} - \frac{t_3(e_i|e_{(i-n)+1}^{i-1})}{\left(\sum_{e\in\xi} c(e|e_{(i-n)+1}^{i-1})\right)^3} \cdots$$

and method X which approximates method P by computing only the first term:

$$\gamma(e_i|e_{(i-n)+1}^{i-1}) = \frac{t_1(e_i|e_{(i-n)+1}^{i-1})}{\sum_{e\in\xi} c(e|e_{(i-n)+1}^{i-1})}$$

However, both of these methods suffer from the fact that when $t_1(e_i|e_{(i-n)+1}^{i-1}) = 0$ or $t_1(e_i|e_{(i-n)+1}^{i-1}) = \Sigma_{e\in\xi} c(e|e_{(i-n)+1}^{i-1})$, the escape probability will be zero (or less) or one respectively. One solution to this problem, suggested by Moffat et al. (1998) and dubbed method AX (for approximate X), is to simply add one to the counts and use the singleton type count in method C.

$$\gamma(e_i|e_{(i-n)+1}^{i-1}) = \frac{t_1(e_{(i-n)+1}^{i-1}) + 1}{\sum_{e\in\xi} c(e_{(i-n)+1}^{i-1}) + t_1(e_{(i-n)+1}^{i-1}) + 1}$$

$$\alpha(e_i|e_{(i-n)+1}^{i-1}) = \frac{c(e_i|e_{(i-n)+1}^{i-1})}{\sum_{e\in\xi} c(e|e_{(i-n)+1}^{i-1}) + t_1(e_{(i-n)+1}^{i-1}) + 1}$$

These methods are based on similar principles to *Katz backoff* (Katz, 1987) one of the more popular techniques used in statistical language processing.

These various escape methods have been subjected to empirical evaluation in data compression experiments. In general, A and B tend to perform poorly (Bunton, 1997; Moffat et al., 1994; Witten & Bell, 1991), while D tends to slightly outperform C (Bunton, 1997; Moffat et al., 1994) and methods based on P (e.g., AX) tend to produce the best results (Moffat et al., 1994; Teahan & Cleary, 1997; Witten & Bell, 1991).

2.3.3 Exclusion

Exclusion (Cleary & Witten, 1984) is a technique for improving the probabilities estimated by PPM. Events which are predicted at higher order contexts do not need to be included in the calculation of lower order predictions. Exclusion of events which have already been predicted in the higher level context will have no effect on the outcome (since they have already been predicted) and doing so reclaims a proportion of the overall probability mass that would otherwise be wasted. Unless explicitly stated otherwise, we shall assume that exclusion is enabled in all models discussed in the remainder of the paper.

2.3.4 Interpolated smoothing

We have discussed the difference between backoff and interpolated smoothing in §2.1 and shown how they can be described within the same framework. While the original PPM algorithm uses a backoff strategy (called *blending*), Bunton (1996, 1997) has experimented with using interpolated smoothing within PPM. The approach is best described by rewriting Equation (2) such that:

$$\alpha\left(e_i \middle| e_{(i-n)+1}^{i-1}\right) = \lambda\left(e_{(i-n)+1}^{i-1}\right) \cdot \frac{count\left(e_i, e_{(i-n)+1}^{i-1}\right)}{count\left(e_{(i-n)+1}^{i-1}\right)}$$

$$\gamma\left(e_i \middle| e_{(i-n)+1}^{i-1}\right) = \left(1 - \lambda\left(e_{(i-n)+1}^{i-1}\right)\right)$$

where:

$$count\left(e_i, e_{(i-n)+1}^{i-1}\right) = \begin{cases} c\left(e_i \middle| e_{(i-n)+1}^{i-1}\right) + k & \text{if } c\left(e_i \middle| e_{(i-n)+1}^{i-1}\right) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$count\left(e_{(i-n)+1}^{i-1}\right) = \sum_{e \in \xi:\, e \text{ is not excluded}} count\left(e, e_{(i-n)+1}^{i-1}\right)$$

and k is the initial event frequency count and a global constant (ideally $k = 0$). The resulting smoothing mechanism is described by:

$$p\left(e_i \middle| e_{(i-n)+1}^{i-1}\right) = \begin{cases} \lambda\left(e_{(i-n)+1}^{i-1}\right) \cdot \dfrac{count\left(e_i, e_{(i-n)+1}^{i-1}\right)}{count\left(e_{(i-n)+1}^{i-1}\right)} + & \text{if } e_{(i-n)+2}^{i-1} \neq \varepsilon \\ \left(1 - \lambda\left(e_{(i-n)+1}^{i-1}\right)\right) \cdot p\left(e_i \middle| e_{(i-n)+2}^{i-1}\right) & \\ \dfrac{1}{|\xi| + 1 - t(\varepsilon)} & \text{otherwise} \end{cases} \tag{7}$$

When using the interpolated smoothing described in Equation (7), it is difficult to ensure that the conditional probability distribution computed sums to one. A simple, though computationally expensive, solution to this problem is to compute the entire distribution and then renormalise its component probabilities such that they do sum to one.

As noted by Bunton (1996, ch. 6), methods A through D may be described using a single weighting function $\lambda : \xi^* \rightarrow [0, 1)$, defined as follows:

$$\lambda\left(e_{(i-n)+1}^{i-1}\right) = \frac{count\left(e_{(i-n)+1}^{i-1}\right)}{count\left(e_{(i-n)+1}^{i-1}\right) + \dfrac{t\left(e_{(i-n)+1}^{i-1}\right)}{d\left(e_{(i-n)+1}^{i-1}\right)}}$$

if we allow the escape method to determine the values of k and a variable $d\left(e_{(i-n)+1}^{i-1}\right)$ as follows:

$$\begin{aligned} &\textbf{A}\text{: } d\left(e_{(i-n)+1}^{i-1}\right) = t\left(e_{(i-n)+1}^{i-1}\right), && k = 0; \\ &\textbf{B}\text{: } d\left(e_{(i-n)+1}^{i-1}\right) = 1, && k = -1; \\ &\textbf{C}\text{: } d\left(e_{(i-n)+1}^{i-1}\right) = 1, && k = 0; \\ &\textbf{D}\text{: } d\left(e_{(i-n)+1}^{i-1}\right) = 2, && k = -\frac{1}{2}. \end{aligned}$$

We note further that method AX may be described within the same framework as follows:

$$\lambda\left(e_{(i-n)+1}^{i-1}\right) = \frac{count\left(e_{(i-n)+1}^{i-1}\right)}{count\left(e_{(i-n)+1}^{i-1}\right) + \dfrac{t_1\left(e_{(i-n)+1}^{i-1}\right)}{d\left(e_{(i-n)+1}^{i-1}\right)}}$$

$$d\left(e_{(i-n)+1}^{i-1}\right) = 1$$

$$k = 0$$

Bunton (1996) observes that the key difference between escape methods A through D is the relative emphasis placed on low order as compared to high order distributions. More emphasis is placed on higher order distributions as both k and $d\left(e_{(i-n)+1}^{i-1}\right)$ increase in numerical value. Thus, while method B places the lowest relative emphasis on higher order distributions, method A tends to place the greatest emphasis on higher order distributions (depending on the value of $d\left(e_{(i-n)+1}^{i-1}\right) = t\left(e_{(i-n)+1}^{i-1}\right)$. Methods C and D fall in between these extremes of emphasis and consistently outperform A and B in data compression experiments.

Blending drops a term of Equation (7) for events which are not novel by assuming that $p\left(e_i \middle| e_{(i-n)+2}^{i-1}\right) = 0$. As we saw in §2.1, this is true of backoff versions of interpolated smoothing methods in general. Bunton notes that, as a consequence, the estimates for novel events are slightly inflated while the estimates for events which are not novel are slightly deflated. Replacing blending with interpolated smoothing remedies this and yields significant and consistent improvements in compression performance (Bunton, 1996, 1997).

2.3.5 Update exclusion

Update exclusion (Moffat, 1990) is a modified strategy for updating the n-gram counts in PPM models. When using the original PPM model with blending and exclusion, the probability of an event which is not novel in a given context, will be estimated in that context alone without blending the estimate with lower order estimates. Update exclusion refers to a counting strategy in which the event counts are only incremented if an event is not predicted in a higher order context. This has the effect that the counts more accurately reflect which events are likely to have been excluded in higher order

contexts. The use of update excluded counts tends to improve the data compression performance of PPM models (Bell et al., 1990; Bunton, 1997; Moffat, 1990).

2.3.6 Unbounded length contexts

One of the goals of *universal* modelling is to make minimal assumptions about the nature of the stochastic processes (or source) responsible for generating observed data. As we discussed in §2.1, *n*-gram models make assumptions about the source to the effect that the probability of an event depends only on the previous $n - 1$ events. Cleary and Teahan (1997) describe an extension to PPM, called PPM*, which eliminates the need to impose an arbitrary order bound. The policy used to select a maximum order context can be freely varied depending on the situation.

A context e_i^j is said to be *deterministic* when it makes exactly one prediction: $t(e_i^j) = 1$. Cleary and Teahan (1995) have found that for such contexts the observed frequency of novel events is much lower than expected based on a uniform prior distribution. As a consequence, the entropy of the distributions estimated in deterministic contexts will tend to be lower than in non-deterministic contexts. Since the event will have occurred at least as many times in the lowest order matching deterministic context as any of the other matching deterministic contexts, it will produce the lowest-entropy probability distribution (Bunton, 1997). Cleary and Teahan (1997) exploit this in PPM* by selecting the shortest deterministic matching context if one exists or otherwise selecting the longest matching context. Unfortunately, the original PPM* implementation provided (at best) modest improvement in compression performance over the original order bounded PPM. When combined with interpolated smoothing and update exclusion, however, PPM* does outperform the corresponding order bounded PPM models in data compression experiments (Bunton, 1997). Furthermore, Bunton (1997) describes an information-theoretic state-selection mechanism which further improves the compression performance of PPM* models.

As noted by Bunton (1997), PPM*'s state selection mechanism interferes with the use of update excluded frequency counts since PPM* does not always estimate the probability distribution using the frequency data from the maximum order matching context. The solution is to use full counts to compute probabilities for the selected context and update excluded counts thereafter for the lower order contexts (see Bunton, 1996, 1997, for further details).

2.3.7 Implementation issues

Since PPM* does not impose an order bound, all subsequences of the input sequence must be stored which makes for increased demands on computational resources. Suffix-tree representations provide a space-efficient means of achieving this end (Bunton, 1996; Larsson, 1996). We have implemented our PPM models as suffix trees using the online construction algorithm described by Ukkonen (1995). The application of this algorithm to the construction of PPM models was first described by Larsson (1996) and the construction developed independently by Bunton (1996) is similar to the Ukkonen–Larsson algorithm in many respects. In addition to being online, these algorithms have linear time and space complexity and, as demonstrated by Bunton (1996), the resulting models have optimal space requirements (in contrast to the original PPM* implementation). Since our suffix trees are constructed from more than one sequence, they are in fact *generalised* suffix trees which require only minor modifications to Ukkonen's suffix tree construction algorithm (Gusfield, 1997). The existence of path compressed nodes in suffix trees complicates the storage of frequency counts and their use in prediction. We have followed the strategies for initialising and incrementing the counts employed by Bunton (1996) to address these complications.

2.4 Long- and short-term models

In data compression, a model which is typically empty initially is constructed incrementally as more of the input data is seen. However, experiments with PPM using an initial model that has been derived from a training text demonstrate that pre-training the model, both with related and with unrelated texts, significantly improves compression performance (Teahan, 1998; Teahan & Cleary, 1996). A complementary approach is often used in the literature on statistical language modelling where improved performance is obtained by augmenting *n*-gram models derived from the entire training corpus with *cache* models which are constructed dynamically from a portion of the recently processed text (Kuhn & De Mori, 1990).

Conklin (1990) has employed similar ideas with music data by using both a *long-term model* (LTM) and a *short-term model* (STM). The LTM parameters are estimated on the entire training corpus and new data is added to the model after it is predicted on a composition-by-composition basis. The STM, on the other hand, is constructed online for each composition in the test set and is discarded after the relevant composition has been processed. The predictions of both models are combined to provide an overall probability estimate for the current event. The motivation for doing so is to take advantage of recently occurring *n*-grams whose structure and statistics may be specific to the individual composition being predicted.

A simple way of achieving the combination of predictions from the LTM and STM is to use a weighted average of the individual predictions (Conklin, 1990). Let $e \in \xi$ be the current symbol to be predicted, M be a set $\{ltm, stm\}$ containing the LTM and STM and $p_m(e)$ be the probability assigned to symbol e by model $m \in M$. The weighted mean of the two predictions is:

$$p(e) = \frac{\sum_{m \in M} w_m p_m(e)}{\sum_{m \in M} w_m} \tag{8}$$

Conklin describes a method for calculating the weights, w_{ltm} and w_{stm} based on the entropies of the distributions generated by the LTM and STM such that greater entropy (and hence uncertainty) is associated with a lower weight. Let P_m be the probability distribution generated by model m. The *relative entropy* of a model is:

$$H_{relative}(p_m) = \begin{cases} H(p_m) / H_{max}(p_m) & \text{if } H_{max}(p_m) > 0 \\ 1 & \text{otherwise} \end{cases}$$

where H and H_{max} are as defined in Equations (3) and (4) respectively. The weight of model m is:

$$w_m = H_{relative}(p_m)^{-b}$$

where $b \in \mathbb{N}$ is a parameter giving an exponential bias towards models with lower relative entropy. Conklin (1990, pp. 70–72) discusses this weighting mechanism in more detail. The combined use of long- and short-term models yields better prediction performance than either the LTM or STM used individually (Conklin, 1990). Finally, Conklin and Witten (1995, p. 61) have used a different scheme, based on the Dempster-Shafer theory of evidence, for combining the predictions of long- and short-term models "with some success" but do not provide any details of the scheme or the performance improvements it yielded.

3 Related work

N-gram models have been used for music related tasks since the 1950s when they were investigated as tools for composition and analysis (see e.g., Brooks Jr. et al., 1957; Hiller & Isaacson, 1959; Pinkerton, 1956). Since extensive reviews of this early research exist (Ames, 1987, 1989; Hiller, 1970), we shall focus here on more recent approaches.

Ponsford et al. (1999), for example, have applied trigrams and tetragrams (without smoothing) to the modelling of harmonic movement in a corpus of 84 seventeenth-century *sarabandes*. The aim was to find out how adequate a simple *n*-gram model would be for the description and generation of harmonic movement in the style. Higher order structure was represented in the corpus through the annotation of events delimiting bars, phrases and entire pieces. A number of pieces were generated from the models and subjected to an informal stylistic analysis. The generated harmonies were "characteristic of the training corpus in terms of harmony transitions, the way in which pieces, phrases and bars begin and end, modulation between keys and the relation between harmony change and metre" (Ponsford et al., 1999, p. 169). The generation of features such as enharmony, which was not present in the corpus, and weak final cadences was attributed mainly to the use of low order models.

The research most closely related to the present work is that of Darrell Conklin (Conklin, 1990; Conklin & Witten, 1995) who used PPM to model the soprano lines of 100 of the chorales harmonised by J. S. Bach. The escape method used was B and both long- and short-term models were employed. The global order bounds of the LTM and STM were set at 3 and 2, respectively, and the predictions combined using a Dempster–Shafer scheme (see §2.4). One of the central features of this work was the representation of multiple attributes, or *viewpoints*, of a melodic sequence. The event space consisted of a cross-product of *basic* viewpoints such as chromatic pitch, onset time, duration and so forth. Viewpoints such as chromatic pitch interval, pitch contour and inter-onset interval were derived from these basic attributes (*derived viewpoints*) and could be combined into *linked viewpoints* which consist of elements of the cross product of the constituent (basic and derived) viewpoints. Finally, this work also introduced *threaded viewpoints* which represent events across larger intervals such as bars and phrases delimited by fermata.[2]

Conklin and Witten (1995) describe a number of *multiple viewpoint systems* consisting of several PPM models trained on different viewpoints whose predictions were combined in the same manner as described in §2.4. Several evaluation techniques were employed. First, split-sample validation (see §4.3) with a training set of 95 compositions and a test set of five compositions was used to compare the performance of different multiple viewpoint systems. The performance measure was the cross entropy (see Equation (5)) of the test set given the model. While a system consisting solely of a viewpoint for chromatic pitch yielded a cross entropy of 2.05 bits per event, more complex multiple-viewpoint systems yielded cross entropy measures as low as 1.87 bits per event. The second means of evaluation was a *generate-and-test* approach similar to that used by Ponsford et al. (1999) from which Conklin and Witten concluded that the generated compositions seemed to be "reasonable" if somewhat normative. Finally, Witten et al. (1994) conducted an empirical study of the sequential chromatic pitch predictions made by human listeners on the same test set of compositions. The entropy profiles derived from the experimental results for each composition were strikingly similar in form to those generated by the model described by Conklin and Witten (1995) – the events about which the model was uncertain also proved difficult for humans to predict.

Hall and Smith (1996) have extended the approach used by Conklin and Witten (1995) to a corpus of 58 twelve-bar blues compositions. The aim was to develop a compositional tool that would automatically generate a blues melody when supplied with a twelve-bar blues harmonic structure. In order to model pitch, zero, first and second order models were

[2] Threaded models are often referred to as *distance n-grams* in the statistical language modelling literature (Huang et al., 1993).

derived from 48 compositions in the corpus. Separate first- and second-order models were derived for each individual chord occurring in the corpus. Rhythm was represented using an alphabet of short rhythmic patterns (e.g., two semiquavers followed by a quaver) and zero, first and second order models were derived from the training set over this alphabet. When generating rhythms, each generated pattern was screened by a set of symbolic constraints for stylistic suitability. The model was evaluated by asking 198 human subjects to judge which of a pair of compositions (of which one was human- and the other machine-composed) was machine-generated. The data consisted of the ten remaining compositions in the corpus and ten compositions randomly selected from the model's output all of which were played to the subjects over a standard harmonic background. Statistical analysis of the results demonstrated that the subjects were unable to distinguish reliably between the human and machine generated compositions.

Reis (1999a) has extended the work of Conklin and Witten (1995) in a different direction through the incorporation of psychological constraints in *n*-gram models. In particular, he argues that storing all *n*-grams (with order less than the global bound) which occur in the data is highly inefficient and unlikely to accurately depict the manner in which humans represent melodies. Reis describes a model which segments the data according to perceptual cues such as contour changes or unusually large pitch or duration intervals. The order of the *n*-grams stored by the model is then determined by the sequence of events back to the previous segmentation point. In the case of ambiguity (e.g., the various segmentation cues do not converge to a single point), all suggested segmentation possibilities are stored. If a novel *n*-gram is encountered during prediction, the distribution delivered by the variable order model is smoothed with a uniform distribution over the alphabet. The model also incorporates perceptually guided predictions for more than one step ahead.

The performance of the model was evaluated on the chorale dataset used by Conklin and Witten (1995) and German folk melodies from the Essen Folk Song Collection (Schaffrath, 1992, 1994) using entropy as the performance metric with a split sample experimental design. The results demonstrated that the model failed to outperform that of Conklin and Witten (1995). In spite of this, Reis's work is useful since it addresses the question of which segmentation and modelling strategies work best when model-size is limited. In particular, he reports the results of an investigation of the predictions of the model when the (perceptually inspired) contexts were shifted. On a set of 205 German folk songs he found that while shifts of between one and ten notes always reduced performance relative to non-shifted contexts, shifts of one note and shifts greater than six produced better prediction than other shifts. Reis suggests that the relatively good performance using single note shifts may be explained by a degree of uncertainty as to exactly where a grouping boundary occurs (i.e., is a large melodic interval included in the preceding group or at the beginning of the following group). The improved performance with longer shifts was attributed to the fact that the average length of the suggested segments was 6.7 notes.

We turn now to more distantly related approaches which we include because they have been used to tackle the same basic task – prediction of an event given a context of immediately preceding events. Assayag, Dubnov and their colleagues (Assayag et al., 1999; Dubnov et al., 1998; Lartillot et al., 2001) have experimented with using an incremental parsing algorithm based on the Lempel–Ziv dictionary compression algorithm (Ziv & Lempel, 1978) in the modelling of musical style. The incremental parsing algorithm adaptively builds a dictionary of sequences as follows. For each new event, it appends the event to the current contender for addition to the dictionary (initially the empty sequence ε). If the resulting sequence occurs in the dictionary, the count associated with that dictionary entry is incremented; otherwise the sequence is added to the dictionary and the current contender is reset to ε. The algorithm then progresses to the next input event. During prediction, an order bound is specified and ML estimated probabilities are used to predict events in a context. When the context does not appear in the dictionary the longest suffix of that context is tried. The IP algorithm has been used successfully, with certain improvements, for the classification of polyphonic music by stylistic genre (Dubnov et al., 1998) and for polyphonic improvisation and composition (Assayag et al., 1999; Lartillot et al., 2001).

Lartillot et al. (2001) have also experimented with another technique for constructing variable length Markov models called *Prediction Suffix Trees* (PST). The algorithm for constructing a PST described by Ron et al. (1996) and used by Lartillot et al. (2001) is offline and operates in two stages: first, a suffix tree is constructed from all subsequences of the input sequence less than a global order bound. Each node in the tree is examined and pruned unless for some symbol in the alphabet, the estimated probability of observing that symbol at the node exceeds a threshold value and is significantly different from the estimated probability of encountering that symbol after the longest suffix of the sequence represented by that state. Lartillot et al. (2001) have derived PSTs from music in a range of different styles and generated new pieces with some success. Triviño-Rodriguez and Morales-Bueno (2001) have derived PSTs to model multiple attributes of the chorale melodies used by Conklin and Witten (1995). They used their models to generate new melodic sequences which have similar statistical properties to the original chorales and which human listeners cannot reliably distinguish from the original chorales.

Mozer (1994) argues that transition table approaches (such as the use of *n*-grams and other Markov models) suffer from two fundamental problems in terms of modelling musical composition: first, an event cannot predict a note that is not its immediate successor without knowledge of the intervening notes; and second, the symbolic representation

used does not facilitate generalisation from one musical context to perceptually similar contexts. In order to overcome these problems, he developed a model based on a recurrent artificial neural network (RANN) and used psychoacoustic constraints in the representation of pitch and duration. When trained and tested on sets of simple artificial pitch sequences with a splitsample experimental paradigm, the RANN model outperformed digram models. However, the results were less than satisfactory when the model was trained on a set of melodic lines from ten compositions by J. S. Bach and used for generation: "While the local contours made sense, the pieces were not musically coherent, lacking thematic structure and having minimal phrase structure and thematic organisation" (Mozer, 1994, p. 273). The neural network architecture appeared unable to capture the higher level structure in these longer pieces of music.

4 Experimental methodology

4.1 Model parameters

A PPM model has been implemented in Common Lisp such that each of the variant features described in §2.3 may be independently selected as parameters to the top-level call. We shall use the following shorthand to refer to each of the model parameters:

Model type: indicated by "LTM" and "STM" for the long- and short-term models respectively while "LTM+" indicates a long-term model in which new data is added to the LTM online as each new event is predicted;[3]

Escape method: indicated explicitly by "A", "B", "C", "D" or "X" (the latter as a shorthand for method AX);

Order bound: indicated by an integer or "*" if unbounded;

Update exclusion: the use of update excluded counts is indicated by "U" – the default does not use update excluded counts;

Interpolated smoothing: PPM's blending is the default while the use of interpolated smoothing is indicated by an "I".

Thus, for example, a PPM long-term model with escape method C, unbounded order, update exclusion enabled and interpolated smoothing is denoted by "LTMC*UI". When it is clear which model is being referred to, we shall, for the sake of readability, drop the model type. When combined with a short-term model with the same parameters, the model would be denoted by "LTMC*UI – STMC*UI" (for readability the two models are separated by a dash). It will be clear that the space of possible parameterisations of the model is very large indeed (even when we limit the range of possible order bounds). As a consequence of this large parameter space, we have applied our techniques incrementally, typically taking the best performing model in one experiment as the starting point for the next.

[3] We have chosen to add new data to the LTM on an event-by-event basis rather than the composition-by-composition basis adopted by Conklin (1990).

4.2 Data

The aim of this research was to assess the performance of PPM variants over a range of different musical styles. The datasets used were all obtained in the **kern format (Huron, 1997) from the *Centre for Computer Assisted Research in the Humanities* (CCARH) at Stanford University, California (see http://www.ccarh.org) and the *Music Cognition Laboratory* at Ohio State University (see http://kern.humdrum.net). During preprocessing, tied notes were collapsed together and the chromatic pitch of each event was converted into a MIDI note number where 60 represents middle C. Each composition therefore consists of a sequence of integers each of which represents a chromatic pitch.

The datasets themselves contain purely melodic music. The first is a collection of 152 folk songs and ballads from Nova Scotia, Canada collected by Helen Creighton between 1928 and 1932 (Creighton, 1966). The dataset was encoded in the **kern format by Craig Sapp and is freely available from the *Music Cognition Laboratory* at Ohio State University. The second dataset used is a subset of the chorale melodies harmonised by J. S Bach (Riemenschneider, 1941). A set of 185 chorales (BWV 253 to BWV 438) has been encoded by Steven Rasmussen and is freely available in the **kern format from CCARH. The remaining datasets come from the Essen Folk Song Collection (EFSC – Schaffrath, 1992, 1994). The collection comprises 6252 (mostly) European folk melodies collected and encoded under the supervision of Helmut Schaffrath at the University of Essen in Germany between 1982 and 1994. A dataset containing all the compositions in the collection encoded in the **kern format is published and distributed by *CCRAH* (Schaffrath, 1995). An additional dataset of 2580 Chinese folk melodies is available on request. The six datasets from the EFSC used in this research contained respectively 91 Alsatian folk melodies, 119 Yugoslavian folk melodies, 93 Swiss folk melodies, 104 Austrian folk melodies, 213 German folk melodies (from dataset kinder) and 237 Chinese folk melodies (from dataset shanxi).

Each dataset is assigned a natural ID as shown in Table 1 and will be referred to henceforth by this ID. Table 1 also contains more detailed information about each dataset, including the number of compositions and events contained in the dataset and the number of chromatic pitches from which the dataset is composed.

4.3 Performance evaluation

Many methods have been used to evaluate the performance of statistical models of music, some of which have been described in §3: the analysis-by-synthesis method used by Hall and Smith (1996) and Triviño-Rodriguez and

Table 1. Melodic datasets used in this research.

ID	Description	No. compositions	No. events	Mean events/composition	|X|
0	Canadian folk songs/ballads	152	8553	56.270	25
1	Chorale melodies	185	9227	49.876	21
2	Alsatian folk songs (EFSC)	91	4496	49.407	32
3	Yugoslavian folk songs (EFSC)	119	2691	22.613	25
4	Swiss folk songs (EFSC)	93	4586	49.312	34
5	Austrian folk songs (EFSC)	104	5306	51.019	35
6	German folk songs (EFSC)	213	8393	39.403	27
7	Chinese folk songs (EFSC)	237	11056	46.650	41

Table 2. The average sizes of the resampling sets used for each dataset.

ID	Training set		Test set	
	Mean no. compositions	Mean no. events	Mean no. compositions	Mean no. events
0	136.8	7697.7	15.2	855.3
1	166.5	8304.3	18.5	922.7
2	81.9	4046.4	9.1	449.6
3	107.1	2421.9	11.9	269.1
4	83.7	4127.4	9.3	458.6
5	93.6	4775.4	10.4	530.6
6	191.7	7553.7	21.3	839.3
7	213.3	9950.4	23.7	1105.6

Morales-Bueno (2001); comparison of human and machine prediction performance (Witten et al., 1994); single-sample Bayesian methods such as Minimum Description Length (Conklin, 1990); and the resampling approach using entropy as a measure of performance as used by Conklin and Witten (1995) and Reis (1999a). We follow the latter approach for two reasons: first, entropy has an unambiguous interpretation in terms of model uncertainty on unseen data (see §2.2); and second, entropy bears a direct relationship with performance in compression and indirectly correlates with the performance of *n*-gram models on practical natural language tasks and is widely used in both these fields (see §2.2). These factors support its use in an application independent evaluation such as this.

Conklin and Witten (1995) used a *split-sample* (or *held-out*) experimental paradigm in which the data is divided randomly into two disjoint sets, a training set and a test set; the *n*-gram parameters are then 14 estimated on the training set and the cross entropy of the test set given the resulting model is computed using Equation 5. Conklin and Witten used a training set of 95 melodies and a test set of 5 melodies. Although commonly used, split-sample validation suffers from two major disadvantages: first, it reduces the amount of data available for both training and testing; and second, with small datasets it provides a biased estimate of the true entropy of the corpus. A simple way of addressing these limitations is to use *k-fold cross-validation* (Dietterich, 1998; Mitchell, 1997) in which the data is divided into k disjoint subsets of approximately equal size. The model is trained k times each time leaving out a different subset to be used for testing and an average of the k cross entropy values thus obtained is then computed.

Since the datasets used are quite small and initial experiments demonstrated a fairly large variance in the entropies computed from different validation sets, we have used 10-fold cross-validation for each dataset in all experiments. The average sizes of these sets are shown in Table 2. In machine learning, differences in model performance as assessed by resampling techniques, such as cross-validation, are often analysed for significance using statistical tests such as the t-test (Dietterich, 1998; Mitchell, 1997). We have followed this approach by comparing the performance of some of our improved models with our emulation of the model developed by Conklin and Witten (1995) as reported in §5.4.

5 Results

5.1 Global order bound and escape method

Our first experiments address the question of how the performance of PPM models is affected by changes in the global order bound. We have tested both the LTM and STM independently with all five escape methods with global order bounds ranging from zero to 14. The results for the LTM and

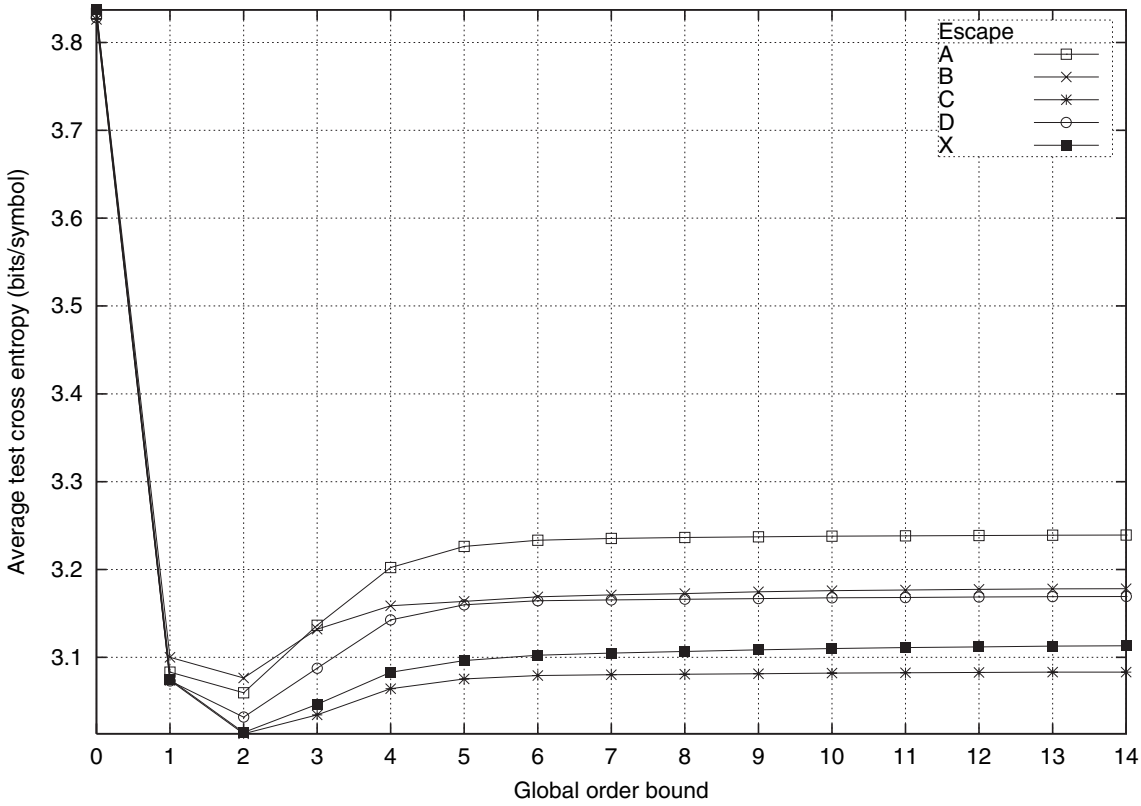

Fig. 1. The performance of the LTM with varying escape method and global order bound.

STM are shown in Figures 1 and 2, respectively. The general *U*-shape of the curves is quite typical; while increasing the global order bound provides the model with more specific contextual information with which to make its predictions, the higher order contexts are also more likely to fail to produce a prediction. Therefore, the model will escape down to lower order models more frequently, thereby wasting more of the probability mass available on apportioning escape probabilities. As the global order bound is increased beyond a certain point this negative influence tends to dominate and performance decreases (Teahan, 1998). Note, however, the relatively shallow worsening of performance of the STM (as compared with the LTM) as the global order bound is increased beyond its optimal value. It seems likely that due to the short length of most of the compositions in the datasets (see Table 1), the models rarely encounter matching contexts longer than about five events and, as a consequence, increasing the global order bound beyond this value has little effect on model performance.

Note from the graphs that, for both the LTM and STM, escape methods A and B perform relatively poorly and escape method C outperforms the others. Methods A and B tended to perform relatively poorly in all the experiments performed, as they have in data compression experiments (Bunton, 1997). As a consequence, they are not considered further in this paper. The optimal global order bound to use is highly dependent on the amount and character of the data being used (Bunton, 1997). As Figures 1 and 2, respectively, demonstrate, the LTM operates best with a global order bound of two, regardless of the escape method used, while the STM performs best with a global order bound of five with escape methods D and C and a global order bound of four with escape method AX.

5.2 Interpolated smoothing and update exclusion

In our next experiments, we investigated the effects that using update excluded counts and interpolated smoothing have on the performance of PPM models with optimal global order bounds as derived in the previous experiment. Thus we have tested the STM and LTM with escape methods C, D and AX with global order bounds of two for the LTM and five (escape methods C and D) or four (escape method AX) for the STM. We have applied the use of update excluded counts and interpolated smoothing to these models both individually and in combination. The results for the LTM and STM are shown in Tables 3 and 4, respectively.

Consider first the results for the LTM shown in Table 3. Perhaps the most striking result is that interpolated smoothing applied in isolation improves performance for all datasets and escape methods. The best performing models on any given dataset use interpolated smoothing in isolation and, as in the previous experiment, escape method C tends on average to outperform methods D and AX. The results for

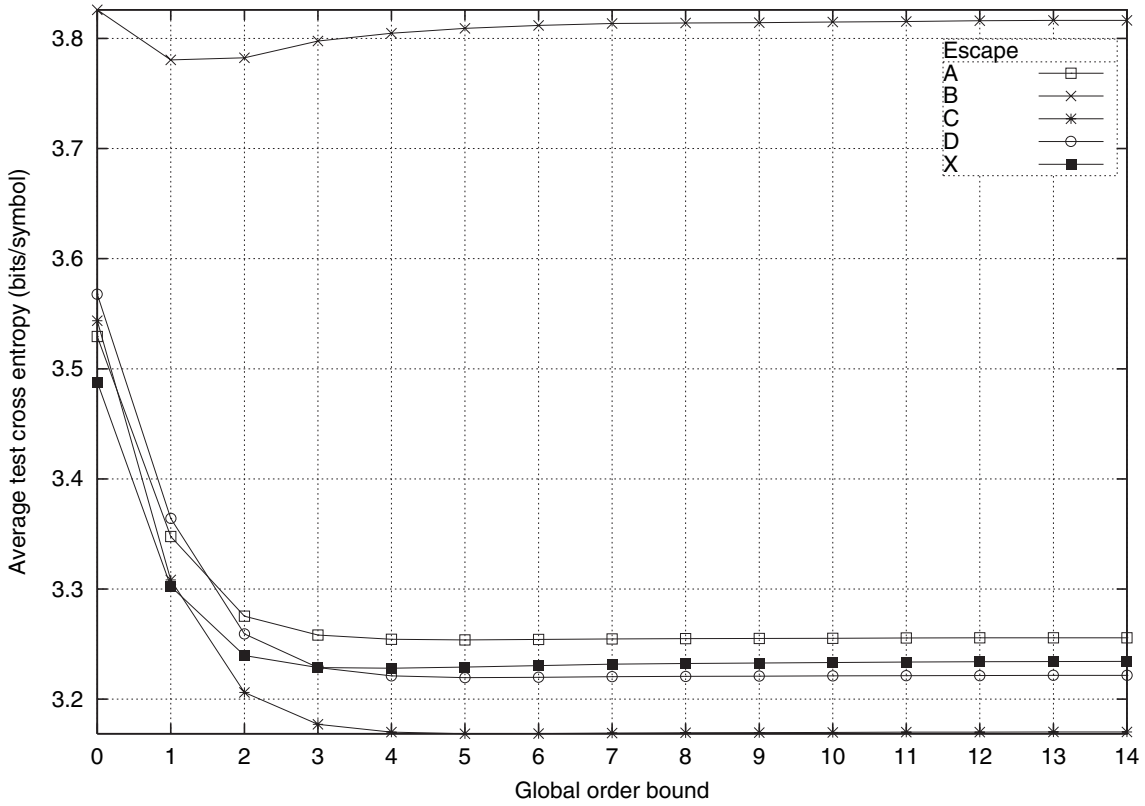

Fig. 2. The performance of the STM with varying escape method and global order bound.

Table 3. Performance of the LTM with a global order bound of two.

Dataset	C2	C2U	C2I	C2UI	D2	D2U	D2I	D2UI	X2	X2U	X2I	X2UI
0	2.933	2.959	2.904	3.127	2.935	2.951	**2.885**	2.967	2.913	2.928	2.887	2.908
1	2.585	2.595	2.563	2.748	2.577	2.581	2.547	2.608	2.557	2.562	**2.544**	2.554
2	3.216	3.204	**3.110**	3.417	3.252	3.208	3.142	3.220	3.207	3.161	3.166	3.129
3	2.882	2.890	2.804	3.179	2.892	2.881	**2.791**	2.954	2.880	2.870	2.824	2.829
4	3.276	3.248	**3.192**	3.483	3.315	3.250	3.220	3.278	3.312	3.231	3.277	3.201
5	3.470	3.480	**3.385**	3.708	3.526	3.485	3.431	3.509	3.518	3.455	3.485	3.429
6	2.620	2.665	2.613	2.897	2.622	2.654	2.599	2.731	2.608	2.642	**2.596**	2.633
7	3.123	3.157	**3.083**	3.423	3.137	3.145	3.094	3.203	3.121	3.123	3.111	3.111
Average	3.013	3.025	**2.957**	3.248	3.032	3.019	2.964	3.059	3.014	2.997	2.986	2.974

update exclusion are, in general, less clear cut. The use of update exclusion alone improves average model performance for escape methods D and AX but not for C (although the margin is small and performance is improved for datasets 2 and 4). The combination of update exclusion and interpolated smoothing tends to impair performance, compared with the performance of models using either technique in isolation, for escape methods C and D; the slight average performance improvement with escape method AX derives from the improved performance on datasets 2, 4 and 5.

Turning now to the results for the STM shown in Table 4, we note that interpolated smoothing applied in isolation tends to improve performance though with less consistency across datasets and escape methods than it does with the LTM. By contrast, update exclusion (applied in isolation) improves average performance when used with escape methods D and AX but impairs performance with escape method C. Even more striking is the finding that the best average performance for each of the three escape methods is obtained using a combination of interpolated smoothing and update exclusion. However, the improvement over models using interpolated smoothing in isolation is much more pronounced for escape methods D and AX than for C where improvement is obtained for datasets 2, 3, 5 and 7 only. The model with best average performance uses escape method AX with update exclusion and interpolated smoothing.

Table 4. Performance of the STM with a global order bound of five (escape methods C and D) or four (escape method AX).

Dataset	C5	C5U	C5I	C5UI	D5	D5U	D5I	D5UI	X4	X4U	X4I	X4UI
0	3.017	3.046	2.988	2.993	3.068	3.048	3.049	2.995	3.081	3.070	3.029	**2.983**
1	3.170	3.209	3.138	3.149	3.218	3.214	3.194	3.153	3.198	3.204	3.162	**3.121**
2	3.120	3.141	3.106	**3.104**	3.175	3.140	3.171	3.107	3.197	3.178	3.156	3.106
3	3.463	3.488	3.466	3.463	3.498	3.491	3.516	3.470	3.440	3.467	3.432	**3.411**
4	3.146	3.178	**3.134**	3.142	3.196	3.176	3.194	3.147	3.214	3.207	3.175	3.139
5	3.264	3.281	3.255	**3.252**	3.316	3.280	3.317	3.257	3.343	3.317	3.303	3.255
6	2.735	2.759	**2.701**	2.706	2.780	2.755	2.755	2.704	2.841	2.856	2.755	2.742
7	3.434	3.437	3.426	3.406	3.504	3.446	3.504	3.417	3.511	3.466	3.485	**3.402**
Average	3.169	3.192	3.152	3.152	3.220	3.194	3.213	3.156	3.228	3.220	3.187	**3.145**

Table 5. Performance of the LTM with unbounded order.

Dataset	C*	C*U	C*I	C*UI	D*	D*U	D*I	D*UI	X*	X*U	X*I	X*UI
0	3.094	3.236	**2.861**	3.234	3.180	3.247	2.930	3.098	3.072	3.153	2.933	2.993
1	2.669	2.843	**2.444**	2.869	2.708	2.839	2.473	2.724	2.648	2.812	2.477	2.651
2	3.336	3.407	**3.115**	3.470	3.454	3.424	3.230	3.308	3.320	3.315	3.230	3.166
3	2.937	3.032	**2.721**	3.188	3.004	3.040	2.761	2.998	2.965	3.028	2.809	2.862
4	3.176	3.199	**3.010**	3.316	3.293	3.205	3.119	3.147	3.263	3.176	3.187	3.056
5	3.515	3.550	**3.340**	3.645	3.665	3.562	3.486	3.488	3.606	3.482	3.542	3.370
6	2.604	2.779	**2.428**	2.926	2.681	2.780	2.468	2.739	2.614	2.748	2.480	2.593
7	3.318	3.449	**3.105**	3.556	3.395	3.434	3.188	3.347	3.298	3.348	3.189	3.205
Average	3.081	3.187	**2.878**	3.275	3.172	3.191	2.957	3.106	3.098	3.133	2.981	2.987

5.3 Comparing PPM and PPM* models

In the next set of experiments, we investigated the effect that the use of update excluded counts and interpolated smoothing have on (unbounded order) PPM* models with a view to comparing the unbounded models with their order-bounded counterparts. As in the previous experiments, we have tested the STM and LTM with escape methods C, D and AX and applied the use of update excluded counts and interpolated smoothing to these models both individually and in combination. The results for the LTM and STM are shown in Tables 5 and 6, respectively, and exhibit broadly similar patterns to the corresponding order bounded results shown in Tables 3 and 4.

Considering first the results for the LTM shown in Table 5, we note that as in the order bounded experiment, interpolated smoothing (applied in isolation) universally improves performance. The use of update exclusion (applied in isolation) universally impairs performance except in combination with escape methods D and AX on datasets 2, 4 and 5. In combination with interpolated smoothing, update exclusion also universally impairs performance except in combination with escape method AX on datasets 2, 4 and 5. The trend for escape method C to outperform the other methods was stronger here than in the order bounded experiment and the best performing model on all datasets used interpolated smoothing and escape method C. Although the use of unbounded orders fails to consistently improve performance when the default blending scheme is used, the combination with interpolated smoothing does lead to consistent performance improvements over the corresponding order bounded models.

Turning now to the results for the STM shown in Table 6, we note that, as in the case of the order bounded STM results, interpolated smoothing applied in isolation tends to improve performance. The effect of update exclusion, both in isolation and in combination with interpolated smoothing, tends to be highly dependent both on the dataset and the escape method used. As in the order bounded experiment, escape methods D and AX tend to combine more fruitfully with update exclusion than method C. The models with best average performance for the former escape methods are obtained with a combination of update exclusion and interpolated smoothing. As in the order bounded experiment, the model with best average performance uses escape method AX with update exclusion and interpolated smoothing and this model outperforms its order-bounded counterpart.

5.4 Combining the long- and short-term models

We now turn to the combined performance of the LTM and STM whose predictions are combined as described in §2.4. In general we have followed an approach in which the best performing models at any given stage are selected for further

Table 6. Performance of the STM with unbounded order.

Dataset	C*	C*U	C*I	C*UI	D*	D*U	D*I	D*UI	X*	X*U	X*I	X*UI
0	3.008	3.046	2.983	2.991	3.060	3.055	3.045	3.000	3.063	3.060	3.020	**2.977**
1	3.170	3.211	3.139	3.150	3.223	3.226	3.201	3.161	3.191	3.194	3.162	**3.117**
2	3.105	3.135	3.097	3.098	3.161	3.144	3.162	3.109	3.168	3.157	3.140	**3.090**
3	3.459	3.491	3.463	3.465	3.495	3.500	3.514	3.477	3.430	3.465	3.427	**3.411**
4	3.136	3.180	**3.126**	3.144	3.186	3.190	3.188	3.158	3.194	3.203	3.165	3.137
5	3.254	3.279	3.248	3.249	3.306	3.286	3.311	3.261	3.317	3.301	3.289	**3.244**
6	2.721	2.753	**2.693**	2.701	2.767	2.759	2.748	2.707	2.814	2.837	2.742	2.731
7	3.432	3.446	3.426	3.414	3.506	3.469	3.508	3.437	3.501	3.467	3.482	**3.406**
Average	3.161	3.192	3.147	3.152	3.213	3.203	3.210	3.164	3.210	3.211	3.179	**3.139**

Table 7. Performance of the best performing LTM, STM and combined models with variable bias.

Dataset	STMC*I	LTMC*I	LTM + C*I	LTM + C*I – STMC*I								
				b = 0	b = 1	b = 2	b = 3	b = 4	b = 5	b = 6	b = 16	b = 32
0	2.983	2.861	2.655	2.495	2.475	**2.468**	2.469	2.474	2.482	2.491	2.564	2.608
1	3.139	2.444	2.375	2.396	2.363	2.347	**2.342**	**2.342**	2.346	2.352	2.412	2.455
2	3.097	3.115	2.712	2.554	2.541	**2.540**	2.548	2.559	2.571	2.584	2.677	2.730
3	3.463	2.721	2.602	2.619	2.597	**2.588**	2.589	2.595	2.604	2.614	2.714	2.791
4	3.126	3.010	2.621	2.484	2.461	**2.454**	2.457	2.465	2.474	2.485	2.560	2.610
5	3.248	3.340	2.833	2.659	**2.649**	2.651	2.661	2.675	2.690	2.706	2.816	2.880
6	2.693	2.428	2.237	2.153	2.120	2.106	**2.102**	2.104	2.109	2.116	2.176	2.212
7	3.426	3.105	2.881	2.694	**2.680**	2.681	2.691	2.705	2.720	2.735	2.841	2.902
Average	3.147	2.878	2.614	2.507	2.486	**2.479**	2.482	2.490	2.500	2.510	2.595	2.648

experimentation. Accordingly, we chose the LTMC*I model for use in these experiments. However, it was found that although an STMC*I model yielded better performance than a STMX*UI model in combination with this LTM even though the latter outperformed the former when used in isolation. This finding in combination with the principle of Occam's razor led us to select an STMC*I model over an STMX*UI model for use in these experiments.

The results of this experiment are shown in Table 7. The first two columns respectively show the performance of the STMC*I and LTMC*I models used in isolation. The third column demonstrates the improved performance afforded by an LTM + C*I model in which events are added online to the LTM as they are predicted (see §2.4). The remainder of Table 7 shows the results obtained by combining the STMC*I model with the LTM + C*I model with a range of different values for the weighting bias *b*. As can be seen, a combined LTM – STM model is capable of outperforming both of its constituent models. The results also demonstrate that optimal average performance is achieved with the bias set to two.

5.5 Overall performance improvements

To illustrate more clearly the performance improvements obtained with the PPM variants discussed in this paper, we have successively applied escape method C, unbounded orders and interpolated smoothing to an emulation of the model used by Conklin and Witten (1995) which is described in our framework as LTMB3 – STMB2 (see §2).[4] The results are shown in Table 8. Paired *t*-tests confirmed the significance of the improvements afforded by incrementally applying escape method C [$t = 31.128$, $df = 79$, $p < 0.001$], unbounded orders [$t = 9.018$, $df = 79$, $p < 0.001$] and interpolated smoothing [$t = 18.281$, $df = 79$, $p < 0.001$]. The tests were performed over all 10 resampling sets of each dataset

[4] At the time of writing, there was insufficient information to enable a precise replication of the experiments described by Conklin and Witten (1995). Any discrepancy between the results reported here for dataset 1 and those of Conklin and Witten (1995) may be attributed to several factors: first, Conklin & Witten used a different set of 100 chorales which is partially disjoint from the set of 185 used here; second, the larger alphabet resulting from the increased size of the dataset; third, the use here of ten-fold cross-validation with an average of 18.5 compositions in the test set compared with the split sample paradigm employed by Conklin and Witten with a training set of 95 and test set of five compositions; and finally, the use of a Dempster–Shafer scheme by Conklin and Witten for combining the predictions of the LTM and STM as compared with the weighted average employed here.

Table 8. Performance improvements to our emulation of the model used by Conklin and Witten (1995).

Dataset	LTM + B3 – STMB2	LTM + C3 – STMC2	LTM + C* – STMC*	LTM + C*I – STMC*I
0	2.905	2.613	2.562	2.468
1	2.676	2.488	2.460	2.347
2	2.997	2.689	2.616	2.540
3	2.934	2.698	2.665	2.588
4	2.974	2.640	2.495	2.454
5	3.233	2.819	2.698	2.651
6	2.555	2.270	2.158	2.106
7	3.111	2.796	2.793	2.681
Average	2.923	2.627	2.556	2.479

(n = 80) results for which are not shown in Table 8. The combined effect of the techniques applied in the LTMC*I – STMC*I model is a 15% improvement in average model performance as measured by cross entropy.

6 Discussion

Before discussing the results presented in §5, some words on the methodology employed are in order. Our goal was to demonstrate that a number of techniques improve the prediction performance of PPM models on monophonic music data. We have approached this task by using cross entropy of the models as our performance metric and applying ten-fold cross validatory resampling on eight monophonic datasets. Since we have been concerned with optimising average performance over all eight datasets, the best performing models selected in some experiments (e.g., the global order bound experiments described in §5.1) will not necessarily correspond to the best performing models on any single dataset. However, these best performing models increase our confidence that the model will perform well on a given dataset without requiring further empirical investigation of that dataset: i.e., we need less information about the dataset to be confident of improved performance.

We have applied the variant techniques incrementally, typically taking the best performing model in a given experiment as the starting point for the next experiment. For example, in §5.4 we took the LTM and STM which yielded best performance independently as the models to combine. Although there is no guarantee that the resulting model reflects the global optimum in the space of possible LTM and STM parameterisations, our aim was to demonstrate that some variant techniques can improve the performance of PPM models and consequently our interest is in the relative, rather than absolute, performance of the PPM variants. In this regard, we have demonstrated that the combined use of three variant techniques affords significant and consistent performance improvements of 15% on average over the model used by Conklin and Witten (1995). We shall now discuss in further detail the implications of the experimental results for each of the variant techniques in turn.

6.1 Escape method

As noted in §2.3.2, there is no principled a priori means of selecting the escape method (the probability to assign to events which have never arisen in a given context before) in the absence of knowledge about the data. In our experiments, escape methods A and B were consistently outperformed by C, D and AX and C fairly consistently outperformed both D and AX (although method AX performed well with the short-term model). These results are broadly in agreement with those obtained in data compression experiments (Bunton, 1996; Moffat et al., 1994; Witten & Bell, 1991). Escape method C is the most commonly used method when Witten-Bell smoothing is used in statistical language modelling (Manning & Schütze, 1999).

6.2 Interpolated smoothing

The use of interpolated smoothing consistently improves model performance (by comparison with PPM's default blending strategy) regardless of the dataset and combination with other variant techniques. This is consistent with results obtained in experiments in data compression (Bunton, 1997) and on natural language corpora (Chen & Goodman, 1999). The reason appears to derive from the fact that backoff smoothing (of which blending is an example) consistently underestimates the probabilities of non-novel events (Bunton, 1997) for which the low order distributions provide valuable information. For natural language corpora, this effect is particularly strong for n-grams with low counts (Chen & Goodman, 1999).

6.3 Update exclusion

While update exclusion generally improves the performance of PPM models in data compression experiments (Bunton, 1997; Moffat, 1990), the results in our experiments were more equivocal. In general, the effects of update exclusion appeared to be highly sensitive to factors such as the dataset, escape method and model type (LTM or STM). In particular, escape methods AX, D and C respectively benefited less from the use of update excluded counts. Furthermore, the

LTM appeared to benefit rather less from update exclusion than did the STM. Finally, when update exclusion did improve average performance, it tended to be the result of improvements on a restricted set of datasets. These findings are not entirely without precedent. The results presented by Bunton (1997) demonstrate that, although it improves average compression performance, update exclusion impairs performance for some of the test files and that escape method C benefits slightly less from the use of update excluded counts than method D.

6.4 Unbounded orders

The use of unbounded orders, as described in §2.3.6, failed to yield consistent improvements in performance for both the LTM and STM except when used in combination with interpolated smoothing. This combination of unbounded orders and interpolated smoothing consistently improves the performance of the best performing order bounded models with interpolated smoothing. These results agree with those obtained in data compression experiments (Bunton, 1997). This is likely to be due to the fact that the optimal order bound varies between datasets. As noted by Bunton (1997, p. 90), order bound experiments "provide more information about the nature of the test data, rather than the universality of the tested algorithms". The advantage of PPM* is that it requires fewer assumptions about the character of the data used.

6.5 Combined LTM and STM

As expected from previous research (Conklin, 1990), combining the predictions of the LTM and STM improves model performance by comparison to that of either model used independently. Curiously, Conklin (1990) found that performance continued improving when the bias b was set to values as high as 128 and greater. In our experiments, the optimal bias setting ranged from one to four depending on the dataset. Further experiments with the bias set to values as high as 32 only yielded further reduction in performance.

7 Summary and conclusions

7.1 Summary

Our goal in this research was to evaluate, in an application independent manner, the performance improvements resulting from the application of a number of variant techniques to a class of n-gram models. Some potential applications of the statistical models we develop are cited in §1 while related work that has been carried out recently with music has been reviewed in §3; we have introduced n-gram modelling in general (§2.1) as well as the information theoretic performance measures that have been used (§2.2). Particular attention was given to PPM models in §2.3, where we described in some detail a number of techniques that have been used to improve the performance on PPM models. These techniques include a range of different escape methods (§2.3.2), the use of update excluded counts (§2.3.5), interpolated smoothing (§2.3.4), unbounded orders (§2.3.6) and combining the predictions of a LTM and STM (§2.4). We have applied these techniques incrementally to eight melodic datasets using cross entropy computed by 10-fold cross-validation on each dataset as our performance metric, as described in §4. The results demonstrated the consistent and significant performance improvements afforded by the use of escape method C (although method AX also performed well with the short-term model), unbounded orders, interpolated smoothing and combining long- and short-term models (see §5). Finally, in §6 we have discussed the results in terms of previous research carried out in the field of data compression and with natural language and music corpora.

7.2 Directions for future research

By way of conclusion, we would like to present some directions that we feel would be profitable to explore in future research. The first set of suggestions concern model development. First, an empirical comparison of the performance of various different techniques for combining the predictions of the LTMand STM, including the weighted average used here and the Dempster–Shafer scheme used by Conklin and Witten (1995), would be useful for future model developers. Second, Bunton (1997) describes an information-theoretic state selection mechanism which replaces the original state selection used in PPM* (see §2.3.6) and which consistently improves performance in data compression experiments. It remains to be seen whether this mechanism can be fruitfully applied with music data. Finally, the extension of the methodology used in this research to comparisons between different modelling approaches could yield interesting results. It would be useful, for example, to compare the performance of the PPM variants analysed here with that of models using other smoothing techniques commonly used in statistical language modelling, such as Katz backoff (Katz, 1987) and Kneser–Ney smoothing (Kneser & Ney, 1995), and models based on the Lempel–Ziv dictionary compression algorithm as used by Dubnov, Assayag and their colleagues (Assayag et al., 1999; Dubnov et al., 1998; Lartillot et al., 2001), the prediction suffix automata used by Lartillot et al. (2001) and Triviño-Rodriguez and Morales-Bueno (2001) and the neural network models described by Mozer (1994).

Our second set of suggestions concern the data used. It should be emphasised that we have restricted our attention to a single attribute of musical sequences: chromatic pitch. None of the conclusions reached in this research can be guaranteed to hold for other attribute domains and representations; we shall need similarly detailed experiments to assess whether the performance improvements recorded here remain valid with these new representations. Therefore, an important consideration is the extension of the approach to

other attributes of musical events and more sophisticated representations of musical works. Conklin & Witten (1995), for example, describe several means of deriving more abstract representations of the musical surface as well as developing methods for combining the predictions of *n*-gram models of these representations (see §2). It is also important to emphasise that our corpora consisted exclusively of folk music; further work is needed to examine the generality of our conclusions in a broader context of musical styles. In a similar vein, we consider it important to extend the approach to homophonic and polyphonic music. The issue of representing such music for training statistical models is discussed by, for example, Assayag et al. (1999), Conklin (2002), Pickens et al. (2003) and Ponsford et al. (1999). Since the results obtained here are in broad agreement with those obtained in data compression and statistical language modelling experiments, we expect the performance improvements to hold some degree of generality and to carry over to these more sophisticated representations of music.

Our final suggestions are methodological. The first concerns the fact that many of the directions cited above concern comparisons between different models. Standard corpora exist for comparing model performance in both the data compression and statistical language modelling communities: e.g., the Calgary corpus (Bell et al., 1990) and LOB corpus (Johansson et al., 1986), respectively. Such standardisation facilitates the objective and empirical comparison of different models and would be highly beneficial to the music processing community. Another methodological issue concerns the validity of entropy as a measure of performance; in order to address this question we need detailed empirical studies of the relationship between entropy measures and model performance on a range of musical tasks such as those outlined in §1. In the meantime, we believe that the techniques described in this paper can be profitably applied to practical musical tasks and that the consequent reduction in cross entropy will translate into actual performance improvement on these tasks.

References

Ames, C. (1987). Automated composition in retrospect: 1956–1986. *Leonardo*, *20*(2), 169–185.

Ames, C. (1989). TheMarkov process as a compositional model: a survey and tutorial. *Leonardo*, *22*(2), 175–187.

Assayag, G., Dubnov, S., & Delerue, O. (1999). Guessing the composer's mind: applying universal prediction to musical style. In: *Proceedings of the 1999 International Computer Music Conference*, (pp. 496–499). San Francisco, CA: ICMA.

Bell, T.C., Cleary, J.G., & Witten, I.H. (1990). *Text Compression*. Englewood Cliffs, NJ: Prentice Hall.

Brooks, F.P. Jr., Hopkins, A.L., Neumann, P.G., & Wright, W.V. (1957). An experiment in musical composition. *IRE Transactions on Electronic Computers*, *EC-6*(1), 175–182.

Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Lai, J.C., & Mercer, R.L. (1992). An estimate of an upper bound on the entropy of English. *Computational Linguistics*, *18*(1), 32–40.

Bunton, S. (1996). *On-Line Stochastic Processes in Data Compression*. PhD thesis, University of Washington, Seattle, WA.

Bunton, S. (1997). Semantically motivated improvements for PPM variants. *The Computer Journal*, *40*(2/3), 76–93.

Chen, S.F., & Goodman, J. (1999). An empirical study of smoothing techniques for language modelling. *Computer Speech and Language*, *13*(4), 359–394.

Cleary, J.G., & Teahan, W.J. (1995). Some experiments on the zero-frequency problem. In: J.A. Storer, & M. Cohn (Eds.), *Proceedings of the IEEE Data Compression Conference*. Washington, DC: IEEE Computer Society Press.

Cleary, J.G., & Teahan, W.J. (1997). Unbounded length contexts for PPM. *The Computer Journal*, *40*(2/3), 67–75.

Cleary, J.G., & Witten, I.H. (1984). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, *32*(4), 396–402.

Conklin, D. (1990). Prediction and entropy of music. Master's thesis, Department of Computer Science, University of Calgary. Available as Technical Report 1990–390–14.

Conklin, D. (2002). Representation and discovery of vertical patterns in music. In: C. Anagnostopoulou, M. Ferrand, & A. Smaill (Eds.), *Proceedings of the Second International Conference of Music and Artificial Intelligence*, (pp. 32–42).

Conklin, D., & Witten, I.H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, *24*(1), 51–73.

Creighton, H. (1966). *Songs and Ballads from Nova Scotia*. New York: Dover.

Dietterich, T.G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*(7), 1895–1924.

Dietterich, T.G., & Michalski, R.S. (1986). Learning to predict sequences. In: R.S. Michalski, J. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: an Artificial Intelligence Approach*, volume II (pp. 63–106). San Mateo, CA: Morgan Kauffman.

Dubnov, S., Assayag, G., & El-Yaniv, R. (1998). Universal classification applied to musical sequences. In: *Proceedings of the 1998 International Computer Music Conference* (pp. 332–340). San Francisco, CA: ICMA.

Ferrand, M., Nelson, P., & Wiggins, G. (2002). A probabilistic model for melody segmentation. In: *Electronic Proceedings of the 2nd International Conference on Music and Artificial Intelligence (ICMAI' 2002)*, University of Edinburgh, Scotland.

Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge, UK: Cambridge University Press.

Hall, M., & Smith, L. (1996). A computer model of blues music and its evaluation. *Journal of the Acoustical Society of America*, *100*(2), 1163–1167.

Hiller, L. (1970). Music composed with computers – a historical survey. In: H.B. Lincoln (Ed.), *The Computer and Music* chapter 4, (pp. 42–96). Cornell, NY: Cornell University Press.

Hiller, L., & Isaacson, L. (1959). *Experimental Music*. New York: McGraw–Hill.

Howard, P.G. (1993). *The Design and Analysis of Efficient Lossless Data Compression Systems*. PhD thesis, Department of Computer Science, Brown University, Providence, USA.

Huang, X., Alleva, F., Hon, H., Hwang, M., Lee, K., & Rosenfeld, R. (1993). The SPHINX-II speech recognition system: an overview. *Computer, Speech and Language*, *2*, 137–148.

Huron, D. (1997). *Humdrum* and *Kern*: selective feature encoding. In: E. Selfridge-Field (Ed.), *Beyond MIDI: The Handbook of Musical Codes* (pp. 375–401). Cambridge, MA: MIT Press.

Jackendoff, R. (1987). *Consciousness and the Computational Mind*. Cambridge, MA: MIT Press.

Johansson, S., Atwell, E., Garside, R., & Leech, G. (1986). The tagged lob corpus. ICAME, The Norwegian Computing Centre for the Humanities, Bergen University, Norway.

Katz, S.M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *35*(3), 400–401.

Kneser, R., & Ney, H. (1995). Improved backing-off for for m-gram language modelling. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1 (pp. 181–184), Detroit, MI.

Kuhn, R., & De Mori, R. (1990). A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(6), 570–583.

Larsson, N.J. (1996). Extended application of suffix trees to data compression. In: J.A. Storer, & M. Cohn (Eds.), *Proceedings of the IEEE Data Compression Conference* (pp. 190–199). Washington, DC: IEEE Computer Society Press.

Lartillot, O., Dubnov, S., Assayag, G., & Bejerano, G. (2001). Automatic modelling of musical style. In: *Proceedings of the 2001 International Computer Music Conference* (pp. 447–454). San Francisco, CA: ICMA.

Manning, C.D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Martin, S., Hamacher, C., Liermann, J.,Wessel, F., & Ney, H. (1999). Assessment of smoothing methods and complex stochastic language modeling. In: *Proceedings of the Sixth European Conference on Speech Communication and Technology* (pp. 1939–1942), Budapest, Hungary.

Mitchell, T.M. (1997). *Machine Learning*. New York: McGraw Hill.

Moffat, A. (1990). Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, *38*(11), 1917–1921.

Moffat, A., Neal, R., & Witten, I.H. (1998). Arithmetic coding revisited. *ACM Transactions on Information Systems*, *16*(3), 256–294.

Moffat, A., Sharman, N., Witten, I.H., & Bell, T.C. (1994). An empirical evaluation of coding methods for multi-symbol alphabets. *Information Processing & Management*, *30*(6), 791–804.

Mozer, M.C. (1994). Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi–scale processing. *Connection Science*, *6*(2–3), 247–280.

Pickens, J., Bello, J.P., Monti, G., Sandler, M.B., Crawford, T., Dovey, M., & Byrd, D. (2003). Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. *Journal of New Music Research*, *32*(2), 223–236.

Pinkerton, R.C. (1956). Information theory and melody. *Scientific American*, *194*(2), 77–86.

Ponsford, D., Wiggins, G.A., & Mellish, C. (1999). Statistical learning of harmonic movement. *Journal of New Music Research*, *28*(2), 150–177.

Reis, B.Y. (1999a). Simulating music learning: on–line, perceptually guided pattern induction of context models for multiple–horizon prediction of melodies. In: *Proceedings of the AISB'99 Symposium on Musical Creativity* (pp. 58–63). Brighton, UK: SSAISB.

Reis, B.Y. (1999b). *Simulating Music Learning with Autonomous Listening Agents: Entropy, Ambiguity and Context*. PhD thesis, Computer Laboratory, University of Cambridge, UK.

Riemenschneider, A. (1941). *371 harmonised chorales and 69 chorale melodies with figured bass*. New York: G. Schirmer, Inc.

Ron, D., Singer, Y., & Tishby, N. (1996). The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, *25*(2–3), 117–149.

Schaffrath, H. (1992). The ESAC databases and MAPPET software. *Computing in Musicology*, *8*, 66.

Schaffrath, H. (1994). The ESAC electronic songbooks. *Computing in Musicology*, *9*, 78.

Schaffrath, H. (1995). The Essen folksong collection. In: D. Huron (Ed.), *Database containing 6255 folksong transcriptions in the Kern format and a 34-page research guide*. Menlo Park, CA: CCARH.

Shannon, C.E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379–423 and 623–656.

Teahan, W.J. (1998). *Modelling English Text*. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand.

Teahan, W.J., & Cleary, J.G. (1996). The entropy of English using PPM-based models. In: J.A. Storer, & M. Cohn (Eds.), *Proceedings of the IEEE Data Compression Conference*. Washington, DC: IEEE Computer Society Press.

Teahan, W.J., & Cleary, J.G. (1997). Models of English text. In: J.A. Storer, & M. Cohn (Eds.), *Proceedings of the IEEE Data Compression Conference*. Washington, DC: IEEE Computer Society Press.

Triviño-Rodriguez, J.L., & Morales-Bueno, R. (2001). Using multi-attribute prediction suffix graphs to predict and generate music. *Computer Music Journal*, *25*(3), 62–79.

Ukkonen, E. (1995). On-line construction of suffix trees. *Algorithmica*, *14*(3), 249–260.

Witten, I.H., & Bell, T.C. (1991). The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, *37*(4), 1085–1094.

Witten, I.H., Manzara, L.C., & Conklin, D. (1994). Comparing human and computational models of music prediction. *Computer Music Journal*, *18*(1), 70–80.

Witten, I.H., Neal, R.M., & Cleary, J.G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, *30*(6), 520–541.

Ziv, J., & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, *24*(5), 530–536.