Simple Bayesian Model for Bitmap Compression *

A. Bookstein (a-bookstein@uchicago.edu)

University of Chicago, 1010 E. 59 St., Chicago, IL 60637, USA

S. T. Klein (tomi@cs.biu.ac.il)

Dept. of Math. & Comp. Sc., Bar-Ilan University, Ramat-Gan 52900, Israel

T. Raita (raita@cs.utu.fi)

Comp. Sci. Dept., University of Turku, 20520 Turku, Finland

Abstract. Bitmaps are a useful, but storage voracious, component of many information retrieval systems. Earlier efforts to compress bitmaps were based on models of bit generation, particularly Markov models. While these permitted considerable reduction in storage, the short memory of Markov models may limit their compression efficiency. In this paper we accept the state orientation of Markov models, but introduce a Bayesian approach to assess the state; the analysis is based on data accumulating in a growing window. The paper describes the details of the probabilistic assumptions governing the Bayesian analysis, as well as the protocol for controlling the window that receives the data. We find slight improvement over the best performing strictly Markov models.

Keywords: IR models, concordances, bitmap compression, Markov modelling

ACM Computing Classification System: E4, G3, H.3.1

* This work was supported, in part, by NSF Grant IRI-9307895-A01 (A.B.), grant 8560195 of the Israeli Ministry of Science (S.K.) and grant 865431 of the Academy of Finland (T.R.). The authors gratefully acknowledge these supports.

1 . Introduction

This paper continues a series of papers (Bookstein et al. 1992, 1994, 1997) that apply advanced statistical models to compress bitmaps. The earliest models assumed that the 1-bits of a bitmap occur independently (Bookstein et al. 1992). In several environments, however, the independence assumption is not valid. For example, consider a concordance built to allow access to a large textual database. In the concordance, each term occurring in the text is associated with a bitmap, and each bit-site to a region of the text obtained by partitioning the text into logically coherent units. The bitmap representing a term has a 1-bit at the positions where the corresponding segment contains the word. Within the text, discussions of specific subjects are clustered, and this imposes a clustering tendency on the 1-bits within the bitmaps belonging to the terms associated with these subjects. Since bit encoding is determined by the probability of its having a given value (for example, using arithmetic coding (Witten et al. 1994)), we expect more accurate modeling to result in improved compression. In this paper we focus on the problem of modeling the clustering rather than on the details of generating codewords.

Our first attempts to represent this clustering involved the use of a variety of Markov Models (Feller 1957). Although significant improvement resulted from using even simple Markov models with a small number of states (Bookstein et al. 1997), ultimately we were limited by the finite memory of such models. We now test an alternative approach, based on the principles of Bayesian statistical analysis (Press 1989), that may correct for this limitation. We continue with the assumption made in our earlier papers that bits are generated within two states, a *cluster* state, which tends to produce a high density of 1-bits, and a *between-cluster* state, which produces 1-bits more sparingly. However, we no longer model the state transitions. Instead, we use Bayesian reasoning to assess the likelihood of being in either state, given the data. The data is gathered within a growing window, which is closed when we decide a state transition was likely. The probability of the next bit taking a value is determined by the data currently in the window, and the bit encoded accordingly.

In the next section we describe, in general, the model on which the remainder of the paper is based. We illustrate these ideas by means of a simple version of the Bayesian model, which we then elaborate. In subsequent sections we describe the protocol used to control the window in which data is collected, as well as describe the experiments used to test the models.

2 . Bayesian Modeling

We conceptualize a bitmap of length D bits as having been generated in a manner similar to that of the hidden Markov model described in Rabiner (1989). At any site in the bitmap, the generator is assumed to be in one of two states: a cluster state $\mathcal{C}$ or a between-cluster state $\mathcal{B}$. These states determine the probability of generating a 1-bit. We diverge from the Markov assumption by no longer modeling state transitions. Rather, we collect data as we scan the bitmap, and use the data to guess the probability of being in either state, and hence the probability of a 1-bit being generated.

Our overall strategy is to encode bits using a dynamic window to accumulate the data. We begin with a window that contains zero bits. As we scan, and encode, bits, the window grows. As we accumulate data in a window, we use Bayesian methods to improve our estimate of the probability of a 1-bit being generated. If at any point we sense a state change, we close the window and begin the process anew.

Suppose then that we are just opening a window. Taking a Bayesian approach, we try to determine the probability distribution of the parameter governing data generation within the window. The parameter here is p, the probability of a 1-bit. We do this by starting with a *prior* distribution for p, based on not having any information, and then improve the distribution as the window grows and more information accrues.

In this problem, the uncertainty in p before developing the window is primarily the result of our not knowing which state we are in, though also, possibly, each state may allow a range of probabilities. Most generally, we assume that in the cluster state $\mathcal{C}$, the distribution of probabilities is given by $f_{\mathcal{C}}(p)$, and that the corresponding distribution in the between-cluster state is denoted by $f_{\mathcal{B}}(p)$. We further assume that θ is the *a priori* probability of being in a cluster state.

When we begin a new window, we have no data on which to base our estimate, and use the prior, unconditional, distribution of p, $P_0(p)$. Given the above assumptions, we have:

$$P_0(p) = \theta \, f_{\mathcal{C}}(p) \; + \; (1-\theta) \, f_{\mathcal{B}}(p). \tag{1}$$

As we accumulate data in the current window, we want to integrate the information it contains to improve our estimate of $P_0(p)$. We denote this density function by $P(p|W)$, indicating that the density function incorporates information about the window W we are scanning. To derive the updated distribution, we use Bayes' formula to get,

$$P(p \mid W) \;=\; \frac{P_0(p) P(W \mid p)}{\int_0^1 P_0(p) \; P(W \mid p) \; dp}. \tag{2}$$

But the probability of the occurrence of a certain window, for a fixed p, is simply

$$P(W \mid p) \;=\; p^{N_1} \, (1-p)^{N_0}, \tag{3}$$

where N_1 and N_0 are, respectively, the number of 1's and the number of zeros in W, which now contains $N_0 + N_1$ values.

We can now estimate the probability of a 1-bit, given the evidence W in the window: $P(1 \mid W) = \int P(1 \mid p, W) P(p \mid W) \; dp$. But since $P(1 \mid p, W) = p$, this integral is simply, $\hat{p}$, the expected value of p, conditional on the evidence in W:

$$\hat{p} \;=\; E(p \mid W)$$

$$= \int_0^1 P(p \mid W)\, p\, dp$$

$$= \frac{\int_0^1 [\theta\, f_{\mathcal{C}}(p) \;+\; (1-\theta)\, f_{\mathcal{B}}(p)]\, p^{N_1+1}(1-p)^{N_0}\, dp}{\int_0^1 [\theta\, f_{\mathcal{C}}(p) \;+\; (1-\theta)\, f_{\mathcal{B}}(p)]\, p^{N_1}(1-p)^{N_0}\, dp}. \tag{4}$$

To proceed, we have to specify the state based distributions. Below we consider two models of increasing complexity.

2.1. Sharp probability distribution

In the simple first model, we assume that the probability is fully determined by the state; that is, we let $p_{\mathcal{C}}$ be the probability of seeing a 1 while in a cluster, and $p_{\mathcal{B}}$ the probability of seeing a 1 while between clusters. Note this allows us to see 1's even if not in the cluster state, and vice versa. Since the probability is determined by the state, we have

$$f_{\mathcal{C}}(p) = \delta(p - p_{\mathcal{C}}),$$

and

$$f_{\mathcal{B}}(p) = \delta(p - p_{\mathcal{B}}).$$

Here $\delta(x)$ is the Dirac δ-function, defined as 0 everywhere but $x = 0$, and whose integral satisfies,

$$\int \delta(x)\, f(x)\, dx = f(0).$$

In other words, $\delta(p - p_{\mathcal{C}})$ is a probability distribution that asserts that $p = p_{\mathcal{C}}$. By equation 1, the prior, unconditional, distribution of p, is given by,

$$P_0(p) = \theta\, \delta(p - p_{\mathcal{C}}) \;+\; (1-\theta)\, \delta(p - p_{\mathcal{B}}),$$

which is a mixture of two Dirac δ-functions.

As we develop the current window, we want to integrate the data it contains, as prescribed by equation 2. We now apply equation 4 to evaluate $\hat{p}$, the expected value of the posterior distribution $P(p \mid W)$:

$$\begin{aligned}
\hat{p} &= \int_0^1 P(p \mid W)\, p\, dp \\
&= \frac{\int_0^1 [\theta\, \delta(p - p_{\mathcal{C}}) \; + \; (1-\theta)\, \delta(p - p_{\mathcal{B}})]\, p^{N_1+1}(1-p)^{N_0}\, dp}{\int_0^1 [\theta\, \delta(p - p_{\mathcal{C}}) \; + \; (1-\theta)\, \delta(p - p_{\mathcal{B}})]\, p^{N_1}(1-p)^{N_0}\, dp} \\
&= \frac{\theta\, p_{\mathcal{C}}^{N_1+1}(1-p_{\mathcal{C}})^{N_0} + (1-\theta)\, p_{\mathcal{B}}^{N_1+1}(1-p_{\mathcal{B}})^{N_0}}{\theta\, p_{\mathcal{C}}^{N_1}(1-p_{\mathcal{C}})^{N_0} + (1-\theta)\, p_{\mathcal{B}}^{N_1}(1-p_{\mathcal{B}})^{N_0}}. \qquad (5)
\end{aligned}$$

Special values

As a check on the reasonableness of the model, it is useful to examine several limiting values.

For example, if $\theta = 0$, we get $\hat{p} = p_{\mathcal{B}}$, and if $\theta = 1$, we get $\hat{p} = p_{\mathcal{C}}$: in such cases, the state is known, and the formula gives the appropriate probability of producing a 1-bit.

We also find that if, as the window grows, the value of N_1 increases indefinitely while the value of N_0 remains fixed, then (assuming $p_{\mathcal{C}} > p_{\mathcal{B}}$) $\hat{p} \to p_{\mathcal{C}}$. That is, as N_1 grows relative to N_0, the evidence becomes increasingly strong that we are in the C state, and the appropriate probability is approached.

Next we find that if $p_{\mathcal{B}} = 0$, we get $\hat{p} = p_{\mathcal{C}}$. While initially surprising, a little thought shows this result is correct. First note that $p_{\mathcal{B}}^{N_1}$, and hence $\hat{p}$, is undefined for $p_{\mathcal{B}} = 0$ unless $N_1 > 0$. But if $p_{\mathcal{B}} = 0$, then if we scan a 1 ($N_1 > 0$), we must be in a cluster state, the only state that permits a 1-bit. Similarly, if $p_{\mathcal{C}} = 1$, $\hat{p}$ is defined only if $N_0 > 0$; but then we must be in a between-cluster state, requiring $\hat{p}$ to be $p_{\mathcal{B}}$, as is indeed the case. Similar comments could be made for the cases $p_{\mathcal{B}} = 1$ and $p_{\mathcal{C}} = 0$, but these are unrealistic given what these probabilities represent.

Finally, if $p_{\mathcal{B}} = p_{\mathcal{C}}$, then $\hat{p} = p_{\mathcal{B}} = p_{\mathcal{C}}$. That is, there is in effect a single state, and the probability of a 1 being generated is the common probability.

2.2. Beta-Distributed Model

If we relax the requirement that the state completely determines the probability parameter, we can create much more flexible models; doing this transcends in principle the *Hidden Markov Model (HMM)* used in our earlier papers. In the model we now examine, we still conceptualize the generator as being in one of two states. The state now *influences*, but doesn't fully *determine*, the probability of generating a 1-bit. We gain this additional flexibility by admitting as probability distributions, density functions $f_{\mathcal{C}}(p)$ and $f_{\mathcal{B}}(p)$ with some dispersion.

The beta-distribution (Johnson et al. 1970) offers a class of models that allows a great deal of flexibility in controlling the shape of the distribution, while at the same time being relatively convenient analytically. We include this model as an example of how the Bayes approach allows us to reach well beyond the Markov framework: we now recognize the possibility that not all clusters are identical, and that in some clusters, the probability of a one-bit is greater than in others. A similar comment could be made for the non-cluster state, although a hybrid model, using a beta-model for the cluster state and a delta-function for the between-cluster state, is possible.

In the beta-distributed model we assume, for parameters $\alpha_{\mathcal{C}} > 0$ and $\beta_{\mathcal{C}} > 0$

$$f_{\mathcal{C}}(p;\ \alpha_{\mathcal{C}}, \beta_{\mathcal{C}}) \ = \ \frac{1}{B(\alpha_{\mathcal{C}}, \beta_{\mathcal{C}})}\, p^{\alpha_{\mathcal{C}}-1}(1-p)^{\beta_{\mathcal{C}}-1},$$

where,

$$B(\alpha_{\mathcal{C}}, \beta_{\mathcal{C}}) \ \equiv \frac{(\alpha_{\mathcal{C}}-1)!\ (\beta_{\mathcal{C}}-1)!}{(\alpha_{\mathcal{C}}+\beta_{\mathcal{C}}-1)!};$$

a parallel distribution defines $f_{\mathcal{B}}(p; \alpha_{\mathcal{B}}, \beta_{\mathcal{B}})$. (Note that here the factorial operator, for non-integer values of its argument, is defined in the usual way in terms of the gamma-function: $x! \equiv \Gamma(x+1)$.)

Substituting the beta-functions for $f_{\mathcal{C}}$ and $f_{\mathcal{B}}$ in equation 4, we find after integration:

$$\hat{p} = \frac{\theta \, \dfrac{B(N_1+\alpha_{\mathcal{C}}+1, N_0+\beta_{\mathcal{C}})}{B(\alpha_{\mathcal{C}}, \beta_{\mathcal{C}})} + (1-\theta) \, \dfrac{B(N_1+\alpha_{\mathcal{B}}+1, N_0+\beta_{\mathcal{B}})}{B(\alpha_{\mathcal{B}}, \beta_{\mathcal{B}})}}{\theta \, \dfrac{B(N_1+\alpha_{\mathcal{C}}, N_0+\beta_{\mathcal{C}})}{B(\alpha_{\mathcal{C}}, \beta_{\mathcal{C}})} + (1-\theta) \, \dfrac{B(N_1+\alpha_{\mathcal{B}}, N_0+\beta_{\mathcal{B}})}{B(\alpha_{\mathcal{B}}, \beta_{\mathcal{B}})}}. \tag{6}$$

The result is analytically complex, and must be evaluated numerically. We must adjust the parameters, $\alpha_{\mathcal{C}}$, $\alpha_{\mathcal{B}}$, $\beta_{\mathcal{C}}$, $\beta_{\mathcal{B}}$ and θ to optimize compression efficiency. However, it is easier to interpret the values of these parameters if we note that the expected value of the Beta-distribution is given by (suppressing the subscripts on the α, β parameters) $E(p) \equiv \bar{p} = \alpha/(\alpha+\beta)$, and the variance by $Var(p) = E(p)(1-E(p))/(\alpha+\beta+1)$, a result reminiscent of the binomial distribution, with α successes, β failures, but $M \equiv \alpha+\beta+1$ tries. It is in terms of $\bar{p}$ and M that the model is most understandingly parameterized, the former indicating the probability of a 1-bit, the latter, our confidence in the parameter. That is, for large M, our model approaches the sharp model described above. Thus, the parameter $\bar{p}_{\mathcal{C}}$ plays a role parallel to that of $p_{\mathcal{C}}$ for the sharp distribution, and similarly for $\bar{p}_{\mathcal{B}}$ and $p_{\mathcal{B}}$.

Experiments will indicate whether the benefit of flexible probabilities within and between clusters justify the extra effort and the need for an extra two parameters.

3 . Window Dynamics

In the preceding section, we have assumed we are in a region where the state is fixed, but that we do not know what the state is. The parameters θ and the beta-distribution parameters allow us to make an information free guess of the likelihood of being in either state, and thereby the probability of the next bit taking the value 0 or 1. We then modify the probability estimate in accordance with the accretion of data as the window grows. In the preceding section, we indicated how to use the data within a window to estimate the probability of a 1-bit being generated. But as the bitmap is generated, we are shifting between states. To take this into account, we must use a protocol that defines how the window grows and when it should be closed.

We begin with a window of size 0, and with the initial probability estimate of eqn 1. Then, as bits are scanned, we upgrade the probability. In principle, we should let the window grow indefinitely. But doing so would put us at a severe disadvantage when the state changes. Thus we introduce a parameter, $W_{\max}$, and when the window-size reaches $W_{\max}$ we stop growing the window. That is, at this point we begin shifting the window, dropping bits at the end while introducing bits at the front. In practice, a finite $W_{\max}$ limits the size of the codeword when a bit inconsistent with the current state is scanned; we choose a value that optimizes performance during compression.

But if at some site an unexpected bit value (or two) is scanned, it suggests that the state may have changed and that the estimate of probabilities should begin again — that is, that the old window may no longer be informative. At this point the old window is closed, and a new one is grown. It may be possible to estimate $W_{\max}$ and the start-over point theoretically.

The ideal way to assess the predictive power of the current window would be to actually look ahead and evaluate how well we are able to compress the *next* several bits (that is, the bits following those already in the window), given the information *within* the window. (The performance expected if the probability prediction mechanism is valid could be computed using information theoretic arguments as discussed in Cover et al. (1991) and Hamming (1980).) If the performance is less than expected, then it indicates we should close the window and begin again. However, the decoder would not be able to follow the same decision procedure, because it does not know the next, incoming bits. This situation is inherent to many adaptive processes (see, e.g. Vitter (1987) and Welch (1984)), and requires that we use a less efficient, delayed update strategy.

The method we adopted is to simulate this strategy retrospectively: after each bit has been encoded, it is appended to the window W and we run a consistency check

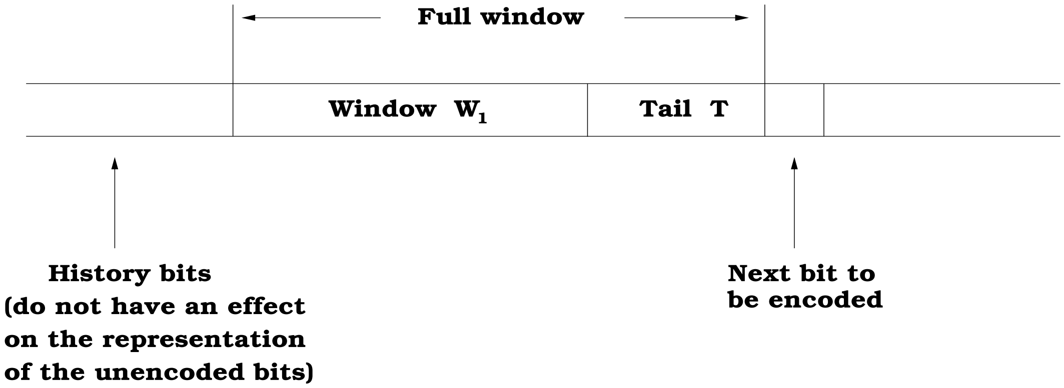

Figure 1. Schema of window structure.

of the bits recently encoded. We backtrack, in turn, one, two, $\ldots b$ bits, where b, the maximum number of bits we can backtrack, is a parameter of our model. Every time we backtrack, we ask whether the bits being reviewed are more consistent with the current window than they are with a new window. If the reviewed bits pass this test for each backtracking step, then we continue, encoding the following bit as an extension of the current window. Otherwise we start a new window. Though the reviewed bits for which the model retrospectively fails have already been encoded as if they were part of the preceding window (which is necessary for the decoder), they are considered as data for the new window. For example, if the first window is closed on the basis of its being inconsistent with the last three bits, those bits act as information for the new window when the next bit is encoded.

We now describe our decision rule for opening a new window. For the purpose of this calculation, after backtracking, we consider the (retrospectively) "current" window, now called W_1, to contain the data up to the point the backtracked data begins; we treat the backtracked data, T, at the tail of the current window as "new" data, available to both the encoder and decoder. This is illustrated is Figure 1.

We examine the data in W_1, and consider whether, with hindsight, we should have continued extending W_1, or begun a new window, W_2. Suppose, looking forward, we see T consists of n 1-bits and m 0-bits. On the basis of W_1 we estimate a probability $\hat{p}_1$ for a 1-bit, while the corresponding probability for W_2 is $\hat{p}_2$. For simplicity, we consider these probabilities to be unchanged as we see successively the bits of T.

We can now use the standard Bayesian argument to estimate the probabilities of W_1 or W_2 being the correct window, given that T follows, where one of the two windows is assumed to be correct. The probability that W_1 is correct is given by:

$$\begin{aligned} P(W_1|m,n) &= \frac{P(W_1)P(m,n|W_1)}{P(W_1)P(m,n|W_1)+P(W_2)P(m,n|W_2)} \\ &= \frac{\omega_1 \hat{p_1}^n (1-\hat{p_1})^m}{\omega_1 \hat{p_1}^n (1-\hat{p_1})^m + \omega_2 \hat{p_2}^n (1-\hat{p_2})^m} \\ &= \frac{1}{1+\frac{\omega_2}{\omega_1}\left(\frac{\hat{p}_2}{\hat{p}_1}\right)^n \left(\frac{1-\hat{p}_2}{1-\hat{p}_1}\right)^m} \end{aligned}$$

for ω_i the probability for W_i. Thus the odds in favor of a new window, $P(W_2)/P(W_1)$ is given by

$$\frac{\omega_2}{\omega_1}\left(\frac{\hat{p}_2}{\hat{p}_1}\right)^n \left(\frac{1-\hat{p}_2}{1-\hat{p}_1}\right)^m. \tag{7}$$

Equation 7, which expresses the evidence infavor of a new window in terms of an odds ratio, can be used directly if the ω's can be properly assessed. This can be tricky. But notice that the ω's are fixed, and the evidence actually influenced by T is fully expressed by the other two factors. Thus we can empirically determine a threshold, and change the window provided the product of the two right-hand factors exceeds this threshold.

It is instructive to rewrite equation 7 as:

$$P(W_2)/P(W_1) = \frac{\omega_2}{\omega_1}\left[\frac{\left(\frac{\hat{p}_2}{1-\hat{p}_2}\right)}{\left(\frac{\hat{p}_1}{1-\hat{p}_1}\right)}\right]^n \left(\frac{1-\hat{p}_2}{1-\hat{p}_1}\right)^{n+m}. \tag{8}$$

Note that the value $n+m$, which is just the number of bits observed while looking ahead, is fixed. Only the term in the brackets actually depends on the evidence, and this, in isolation, can serve as a measure of how strong is the claim to start a new window. The factor $[(1-\hat{p}_2)/(1-\hat{p}_1)]^{n+m}$ is the value of the odds favoring a new window, provided all the bits being tested are zero. Suppose this would result in one of the windows, say W_i, being selected. Each 1-bit then modifies this by a factor of the odds ratio $(\hat{p_2}/(1-\hat{p_2}))/(\hat{p_1}/(1-\hat{p_1}))$. If collectively this doesn't compensate for the initial weight, we use W_i, else we use the other window.

To estimate p_1 we used the last probability estimate made from the evidence in W_1 before the new evidence was evaluated. A few reasonable options are available for estimating p_2. For example, we can use the *a priori* probability, to assess whether the evidence in the window is at this point any better than no evidence at all. An opposite bias can be obtained by estimating p_2 from the actual evidence itself, to see if this high-value estimate is sufficiently larger than that based on W_1 to justify opening a new window.

In principle, the window update procedure could be applied recursively to the resulting tail T, perhaps allowing an even better (and shorter) basis for estimating the probabilities of the upcoming bits. However, since we restricted the length of T just to 6 bits at most, the recursive procedure was not applied in the experiments.

We next illustrate the window update protocol by an example. Suppose we just encoded bit b_{511}, and that W now consists of bits $b_{500}, b_{501}, \cdots, b_{511}$. Next we must test for the goodness of the window. We do this in stages:

First stage: For our first consistency check, W_1 will consist of bits $b_{500}, b_{501}, \cdots, b_{510}$ and T of the single bit, b_{511}. On the basis of W_1, we compute $\hat{p_1}$, the probability of a 1-bit, given W_1, and the size of the codeword for b_{511}. Get the corresponding quantity for an empty window, and call it $\hat{p_2}$. Evidence for closing the window is given by $\hat{p_2}/\hat{p_1}$ if b_{511} is 1, else it is given by $(1-\hat{p_2})/(1-\hat{p_1})$. This is equation 7, ignoring the ω's. We close the window if the ratio is larger than some threshold value γ, beginning a new window with b_{511}. If the ratio is smaller than γ, we go to next stage.

Second stage: First we backtrack, redefining W_1 as $b_{500}, b_{501}, \cdots, b_{509}$ and T as b_{510}, b_{511}. We compute $\hat{p_1}$, the probability of a 1-bit, given the reconstituted W_1; $\hat{p_2}$ again is the probability of a 1-bit given the empty window. We now use equation 7 to get the weight for a new window, based upon the evidence of T. If the evidence is large enough, we close the current window, and repeat the process beginning with the window consisting of bits b_{510}, b_{511}. Else we continue with the third stage (which we suppose, for this example, to be the last).

Stage three: We now compute $\hat{p_1}$ on basis of the new $W_1 = (b_{500}, \cdots, b_{508})$, and $\hat{p_2}$, assuming the null window. We then compute the weight favoring a new window based on the evidence of the new T, consisting of the bit sequence $b_{509}, b_{510}, b_{511}$, as given by equation 7. If the weight supports opening a new window, we close the current window, and begin the cycle again starting with the window made up of bits $b_{509}, b_{510}, b_{511}$. Else we accept the old window, make bits $b_{501}, \cdots, b_{511}$ the current window, and continue. If additional stages are desired, we continue as above.

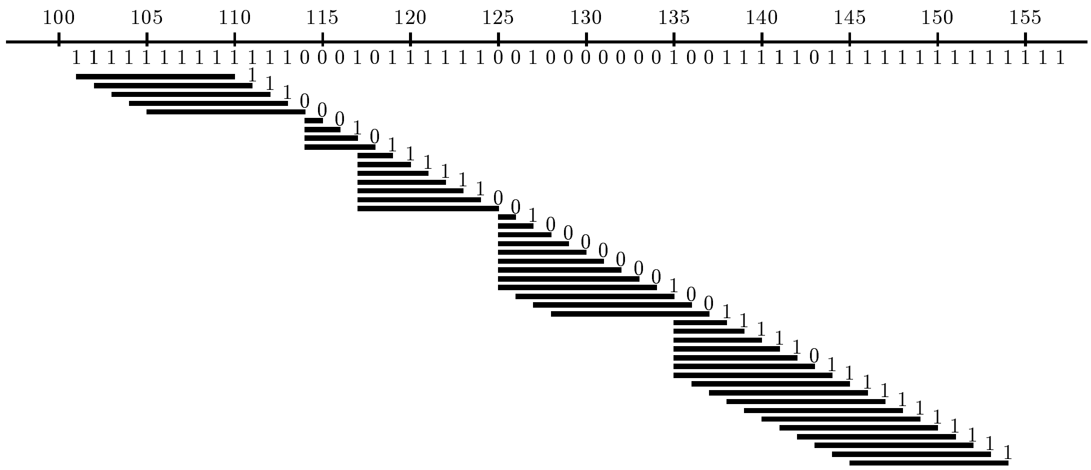

Figure 2. Example of window development.

An actual detailed example of window development, taken from the experiment described below, appears as Figure 2. The numbers on top of the figure denote bit positions of the specific bitmap we are studying and the actual bits are shown beneath. The maximal length of the window, W_{max}, is set to 10 and the figure depicts the window dynamics during the encoding of bits $111 \ldots 155$. At the start, the window contains the 1-bits at indices $101 \ldots 110$. This run of 1-bits gives a high probability for the upcoming 1-bit at index 111. The bit is encoded and inserted into the window. In the update procedure, we notice that the window W_1 is always a better predictor for the corresponding tail T than an empty window, so only one change is made: we discard the oldest bit from the window to prevent an overflow. For the next two bits at indices 112 and 113, the same is repeated. When the 0-bit at index 114 has been encoded, we note that the threshold value used in the consistency check allows one spurious bit inside a run of 1-bits, so again, only the oldest bit is discarded. However, the next 0-bit gives already enough evidence about the end of the run of ones. This is noticed at stage two of the update process: the eight 1-bits in W_1 are not consistent with the pair of zeroes in the tail T, so we open a new window and initialize it with the contents of T. In this way, the update procedure captures the essential features of the bit sequence: if there are only a few 0-bits in a 1-bit cluster (or vice versa), we are satisfied with it, since the overall tendency is clear. Also, when we are scanning a truly heterogeneous bitmap region, there is no reason to change the window radically (see e.g. the situation when the encoding of the bit at index 144 commences). This implies that the change of the state is performed only when necessary, i.e. at the boundaries where the characteristics of the bit string change considerably.

4 . Experiments

In order to compare the new technique with previous methods, we used the same test databases as in our earlier papers, namely: the King James Version of the Bible in English (its $D = 929$ chapters acting as documents); and a subset of the *Trésor de la Langue Française* (TLF). The TLF is a database of 680 MB of French language texts (112 million words) of the 17^{th}–20^{th} centuries (see Bookstein et al. (1992) for more details of the collection), whose uncompressed concordance spans about 345 MB (excluding references to the 100 most frequent words, considered as *stop-words*). The subset used in this study consisted of the the 35070 terms belonging to the (lexicographic) range between `elle` and `flaube`.

The models we are developing are intended for terms that show a considerable degree of clustering. But clustering strength is determined both by the nature of the word and the choice of what we define as a document, which is represented as a single bit in our bitmaps. In our investigation of the TLF, we chose as the document the level immediately below the "book" level in the TLF hierarchy, resulting in bitmaps of length $D = 38757$ bits. The chosen unit is convenient for the construction of the concordance, but may obscure some of the underlying clustering because of its large size.

Our first task is to estimate the parameters which give the best compression, that is, which minimize the function $-\sum_{i=1}^{D} \log \hat{p}_i$. We here use the fact that we can encode a bit with $\log \hat{p}_i$ bits, if the encoding is based on $\hat{p}_i$ being the probability that the i-th bit of the bitmap takes the value it does; this probability is determined by the evidence in the window.

Performing this minimization analytically is hard, so we reverted to search methods that may yield sub-optimal parameter values. The parameters divide into two sets: the model parameters (θ, $p_{\mathcal{B}}$, and $p_{\mathcal{C}}$ for the sharp model); and three window parameters: the maximum window size $W_{\max}$, the maximal backtracking length b in the update process, and the threshold γ that controls when we should restart the window. For the Beta-model, we need θ, $\bar{p}_{\mathcal{B}}$, $\bar{p}_{\mathcal{C}}$, $M_{\mathcal{B}}$ and $M_{\mathcal{C}}$. (Recall that the α's and β's of the Beta distribution can be expressed in terms of the $\bar{p}$'s and M's in a straightforward way.)

To simplify our task of finding optimal values for our parameters, we note that, if n_1 is the number of 1-bits in the map, the sharp model requires that:

$$\frac{n_1}{D} = \theta\, p_{\mathcal{C}} \; + \; (1-\theta) p_{\mathcal{B}},$$

yielding as the value for θ,

$$\theta = \frac{n_1/D - p_{\mathcal{B}}}{p_{\mathcal{C}} - p_{\mathcal{B}}}.$$

For the Beta-model, there is no fixed probability for bit generation, but we can use the above formula, with $(\bar{p}_{\mathcal{C}}, \bar{p}_{\mathcal{B}})$ substituted for $(p_{\mathcal{C}}, p_{\mathcal{B}})$ as an initial estimate of θ. It reasonable to expect that n_1/D, the frequency of occurrence of 1-bits, satisfies the constraint $\bar{p}_{\mathcal{B}} \leq n_1/D \leq \bar{p}_{\mathcal{C}}$; this implies that $0 \leq \theta \leq 1$.

For each set of terms tested, we chose manually several combinations of parameter values obeying the restrictions above, and then searched for local optima. The large number of optimization runs performed showed, that the overall shape of the function is 'rippling' and with high probability, it does not have large minima or maxima. Due to this, we can safely fix most of the input parameters without a notable loss in compression efficiency. Some tuning may be appropriate, however, if the 1-bit densities or clumping properties change considerably from bitmap to bitmap.

Table 1 gives descriptive statistics for the tested files and summarizes the compression performance of the various models tested. For the Bible, only words appearing in at least 60 chapters, and thus occurring in between 60 and 929 chapters, were considered; for TLF, the terms (bitmaps) were partitioned into three classes according to the density of 1-bits. The threshold values were 78, 388 and 1162.

Table I. Statistics and compression results

Database:	Bible	TLF		
	60–929	78–387	388–1161	1162–38757
terms	623	2032	619	381
occurrences	131874	352522	402890	1387698
independent	2.683	9.09	7.26	4.02
best 4-state	2.544	8.48	6.65	3.48
HMM	2.490	—	6.62	3.39
Bayes-Beta	2.523	8.38	6.55	3.42
Bayes-Sharp	2.556	8.44	6.65	3.52

The upper part of Table 1 shows, for each class, the number of different terms, and their total number of occurrences. The lines of the lower part correspond to various bitmap compression methods. The independence model is the 1-state Markov model used in Bookstein et al. (1992); this is a very simple model that does not take clustering into account and therefore performs relatively poorly. In Bookstein et al. (1997) we studied the performance of different Markov models; the values cited here correspond to choosing, for each class, the model giving the best compression among all traditional 4-state Markov models. The line entitled HMM gives the compression figures of Bookstein et al. (1997) for the hidden Markov model. The omission of a HMM result is due to the enormous execution time needed for the run to produce this figure. The last lines correspond to the Bayesian models described herein.

To understand the values in the table, recall that we are representing the top level of the concordance as follows: for each term, we list sequentially the documents in which the term occurs. This list can be conceptualized as a bitmap, D bits in length, with a 1-bit for each document in which the term occurs. As our measure of compression for the list corresponding to a term, we compute the number of bits needed to encode this list with our methods, and divide this value by the number of documents in which the term occurs. The table gives the average of this quantity for all the terms in a class. In other words, it is the average, per 1-bit within an uncompressed bitmap, of the number of bits needed to encode the 1-bits of all the bitmaps in this class.

As in our earlier studies (Bookstein et al. (1997)), we have not included the cost of storing the necessary parameters, which is negligible in most of the cases. Since it seems that varying $W_{\max}$ and the threshold value γ does not have a large effect on the results, they were fixed throughout the experiments, leaving only three or five parameters for the sharp and Beta models, respectively, as compared to four parameters for each of the HMM and 4-state models. Since only the high frequency terms are to be compressed, the average number, per term occurrence, of added bits caused by this overhead, is very small.

The parameter values $\bar{p}_{\mathcal{B}}, \bar{p}_{\mathcal{C}}, \sigma_{\bar{p}_{\mathcal{B}}} = \sqrt{Var(\bar{p}_{\mathcal{B}})}, \sigma_{\bar{p}_{\mathcal{C}}} = \sqrt{Var(\bar{p}_{\mathcal{C}})}$ define the generating model. For about 45% of the bitmaps we were able to set θ equal to zero, implying that a single beta-distribution suffices for the majority of terms. Note, however, that this does not invalidate an underlying multi-state generating process. Rather, the intrinsic variability of the beta-distribution seems adequate to represent the impact of the various states.

For the other 55% of the terms, it was useful to incorporate the services of a second beta-distribution. However, for both the cluster and between cluster distributions, the probability of a 1-bit was low: the average value for $\bar{p}_{\mathcal{B}}$ was 0.005 and for $\bar{p}_{\mathcal{C}}$ 0.144. The average standard deviations were 0.010 and 0.096, respectively. Thus, for these cases, the spread was comparable to the size of the values themselves.

As can be seen, in all our tests, the Bayesian model outperformed the best 4-state models by 1–2%. Surprisingly, the new approach even improved upon the compression obtained by the highly CPU-intensive HMM model in one of the cases. We thus conclude that the Bayesian technique described in this paper gives a good time/space tradeoff, compressing better than the faster 4-state models, and using significantly less processing time than the HMM.

5 . Conclusion

Bitmaps can be very convenient when representing concordances. But in realistic applications, they are large, and storing them is expensive. The simplest techniques for compressing bitmaps, based on models that represent bitmap generation as a sequence of independent events, in fact work quite well. But when dealing with large databases, even a small percent improvement in cost can yield large absolute benefits. Our investigations were directed to those applications for which the size of the database justifies the additional complexity.

As is typical in the design of optimal systems, we found that added complexity tends to improve overall performance, but with each increment in complexity yielding diminishing returns. For example, the best 4-state models in fact come quite close to the performance of the Hidden Markov Models, which are very flexible, and very expensive, models that we considered as a bound on our ability to improve performance using the route of simple probabilistic models.

In this paper, we were interested in assessing whether alternatives to Markov modeling, that relieved the constraint of the Markov model's short memory, could match or improve performance. We used an underlying state model, but applied Bayesian reasoning, rather than Markov transition probabilities, to assess the state. Our results were comparable to that of the HMM, and at comparably little cost.

The convergence towards the performance of the HMM, and the agreement of two rather different approaches on two very different databases, leads us to expect that striking compression improvement is unlikely without a radical increase in model complexity. However, we examined only one application — that of representing the concordance of a textual database. Other applications, in which clustering is more pronounced, may lead to quite different results, and for which the variety of models we developed could be more strongly differentiated.

6 . References

Bookstein A, Klein S.T and Raita T (1992) Model based concordance compression. In: Storer J.A and Cohn M, eds. *Proc. Data Compression Conference DCC–92*, Snowbird, Utah, pp. 82–91.

Bookstein A, Klein S.T and Raita T (1994) Markov models for clusters in concordance compression. In: Storer J.A and Cohn M, eds. *Proc. Data Compression Conference DCC–94*, Snowbird, Utah, pp. 116–125.

Bookstein A, Klein S.T and Raita T (1997) Modeling word occurrences for the compression of concordances. ACM Transactions on Information Systems, 15:254–290.

Bookstein A, Klein S.T and Ziff D.A (1992) A systematic approach to compressing a full text retrieval system. Information Processing and Management, 28:795–806.

Cover T.M and Thomas J.A (1991) Elements of Information Theory. John Wiley & Sons, New York.

Feller, W (1957) An Introduction to Probability Theory and Its Applications, vol I. John Wiley & Sons, New York.

Hamming, R. W (1980) Coding and Information Theory. Prentice-Hall, Englewood Cliffs, NJ.

Johnson, N.L and Kotz, S (1970) Distributions in Statistics: Continuous Univeriate Distributions–2. Wiley, New York.

Moffat, A and Stuiver, L (1996): Exploiting Clustering in Inverted File Compression, In: Storer J.A and Cohn M, eds. *Proc. Data Compression Conference DCC–96*, Snowbird, Utah, pp. 82–91.

Press, J.S (1989) Bayesian statistics : principles, models, and applications. Wiley, New York.

Rabiner L.R (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE, 77:257–286.

Vitter J.S (1987) Design and analysis of dynamic Huffman codes. Journal of the ACM, 34:825–845.

Welch T.A (1984) A technique for high performance data compression. IEEE Computer, 17:8–19.

Witten I.H, Moffat A and Bell T.C (1994) Managing Gigabytes, Compressing and Indexing Documents and Images. International Thomson Publishing, London.

Address for Offprints: Timo Raita

Dept. of Computer Science

University of Turku

SF-20500 Turku

Finland