Chapter 2

Language Modelling

2.1 Introduction

In a natural language, some phrases will be more common than others. To take a canonical example from the speech recognition community, the phrases "we recognise speech" and "we wreck a nice beach" are acoustically very similar, but the former is intuitively much more probable than the latter. In a speech recognition system, the use of a language model which is able to capture this intuition is of great help when noise is present in the recorded signal.

More formally speaking, the goal of a language model is to provide a probability distribution over strings, $(t_1, \ldots, t_M)$, where the individual elements, t_i, are drawn from a set of tokens, $\mathcal{T}$. For example, the set of tokens may be the letters, numerals, punctuation symbols, and whitespace elements used in written text in a particular language. In this case, the tokens will be referred to as *symbols*, and the set of all possible symbols will be referred to as the *alphabet*, written as $\mathcal{A}$. To give another example, the set of tokens might be a set of words taken from a natural language. In this case the tokens are referred to as *terms* and the set itself is the *vocabulary*, $\mathcal{V}$. The same approaches may be generalised to other domains such as genetic information or the state sequence of some abstract finite state machine although for purposes of illustration the discussion in this chapter is restricted to the first two examples only.

Given such a model, it is often useful to be able to predict the next token conditioned on the string seen so far. This is simply a matter of using the conditional distribution

$$\Pr(t_{M+1} \mid t_1, \ldots, t_M) = \frac{\Pr(t_1, \ldots, t_{M+1})}{\Pr(t_1, \ldots, t_M)} \tag{2.1}$$

$$= \frac{\Pr(t_1, \ldots, t_{M+1})}{\sum\limits_{t_{M+1}} \Pr(t_1, \ldots, t_{M+1})} \tag{2.2}$$

Putting this another way, the distribution over full strings can be broken down sequentially,

$$\Pr(t_1, \ldots, t_M) = \prod_{i=1}^{M} \Pr(t_i \mid t_1, \ldots, t_{i-1}) \tag{2.3}$$

This decomposition is exact as long as all preceding tokens are taken into account when evaluating the predictive distribution, $\Pr(t_i \mid t_1, \ldots, t_{i-1})$. While it is without doubt true that long range correlations play some role in natural languages, for the sake of mathematical and computational simplicity models are often restricted to considering only the previous $N - 1$ tokens, for some fixed N. In other words,

$$\Pr(t_i \mid t_1, \ldots, t_{i-1}) \approx P_N(t_i \mid t_{i-N+1}, \ldots, t_{i-1}) \tag{2.4}$$

This assumption results in a *finite context model*. The string $(t_{i-N+1}, \ldots, t_{i-1})$, represented $\mathcal{C}_i^N$, is referred to as the *context* of token i. The superscript N will be omitted for the sake of clarity if confusion is unlikely. A sequence of N tokens is often referred to as an N-gram. Typical values of N are 5 or 6 for modelling at the level of letters, and 3 for modelling at the level of words, where the token set is typically much larger.

This chapter considers language modelling from a Bayesian viewpoint. The goal of the chapter is to offer new interpretations of existing models. It is hoped that this will help develop an understanding for the relative successes of some models and to ultimately act as a guide in the creation of new models.

2.2 Measuring language model performance

In order to successfully investigate a language model it is important to be able to quantitatively measure its performance. The metric used in this thesis is *information rate*, defined as

$$R_I = -\frac{1}{M} \sum_{\boldsymbol{t}} P(\boldsymbol{t}) \log_2 Q(\boldsymbol{t}) \tag{2.5}$$

where the sum is over strings of length M, P is the true distribution of the language and Q is the distribution whose performance is being evaluated. Typically the expectation will be approximated using a small number of sample strings, or in many cases just a single typical example. Information rate is measured in bits per symbol[1], and is equal to the compression rate which can be achieved by an ideal arithmetic coder using the language

[1]This depends on the base of the logarithm, using a natural logarithm results in a measurement in *nats*

model. Expanding (2.5),

$$
\begin{aligned}
R_I &= -\frac{1}{M}\sum_{\boldsymbol{t}} P(\boldsymbol{t}) \log_2 \frac{Q(\boldsymbol{t}) \cdot P(\boldsymbol{t})}{P(\boldsymbol{t})} && (2.6)\\
&= \frac{1}{M}\sum_{\boldsymbol{t}} P(\boldsymbol{t}) \log_2 \frac{P(\boldsymbol{t})}{Q(\boldsymbol{t})} - \frac{1}{M}\sum_{\boldsymbol{t}} P(\boldsymbol{t}) \log_2 P(\boldsymbol{t}) && (2.7)\\
&= D_{KL}(P, Q) + H(P). && (2.8)
\end{aligned}
$$

The first term is the Kullback Leibler (KL) divergence [46] between the distribution of the language model and the true distribution, and is provably non-negative, with a minimum value of zero occurring only at the point where P and Q are equal. The second term is the entropy of the true distribution, which therefore provides a lower bound for the information rate, achieved only for a language model whose distribution matches the true distribution of the language exactly. A famous experiment by Shannon [86] used guesses made by human subjects to estimate the entropy of the English language (ignoring capitalisation and punctuation). The figure obtained was between 0.6 and 1.3 bits per symbol.

An alternative metric is the perplexity, which can be thought of as the number of symbols which must be chosen between in the predictive distribution (this is exact in the case of a uniform distribution over a subset of the symbols). Perplexity is directly related to the information rate,

$$\text{Perplexity} = 2^{R_I} \tag{2.9}$$

The non-linear relationship between the two means that small gains in information rate can appear artificially larger when quoted in terms of perplexity if the baseline performance of the model is poor. Information rate will be used exclusively in this thesis as it is more directly related to experimentally measurable quantities.

2.3 Review of existing techniques

The problem of language modelling has a long history, and has developed to some extent in parallel in different fields. As a result, there exists a wide variety of different approaches, and in some cases the same or very similar approaches are known by different names. The following sections introduce some of the more widely used approaches, highlighting some of the relationships between them. The list is however not exhaustive, and many other approaches exist in the literature. For further background, see [18] and [79], as well as references therein.

2.3.1 Maximum likelihood estimation

One way of interpreting the conditional distribution used in the sequential prediction task as described in (2.3) and (2.4) is as a set of discrete distributions over the token set, one for each possible context. In practice there are strong correlations between predictions made in different contexts, which will be the subject of a later section. However, one of the most straightforward approaches to the language modelling problem is to treat these distributions as being independent of each other.

The distributions themselves can be learned from example data. The simplest approach is to use a maximum likelihood estimate of the predictive distribution. In other words, let $P(t \mid \mathcal{C})$ be the predictive distribution in context $\mathcal{C}$. The maximum likelihood estimate is simply given by

$$P(t \mid \mathcal{C}) = \arg\max_{P} \prod_{i=1}^{M} P(t_i \mid \mathcal{C}_i) \tag{2.10}$$

It is often more convenient to maximise the logarithm of the probability,

$$P(t \mid \mathcal{C}) = \arg\max_{P} \sum_{i=1}^{M} \log P(t_i \mid \mathcal{C}_i) \tag{2.11}$$

which results in an identical estimate of $P(t \mid \mathcal{C})$. Introducing p_{jk} as shorthand notation for $P(t = t_j \mid \mathcal{C}_k)$, and $\{\lambda_k\}$ as Lagrange multipliers to enforce the constraints that for each k, $\sum_j p_{jk} = 1$, this optimisation can be performed in the usual way:

$$\begin{aligned} 0 &= \frac{\partial}{\partial p_{jk}} \left(\sum_{i=1}^{M} \log P(t_i \mid \mathcal{C}_i) + \sum_{k'} \lambda_{k'} \sum_{j'} p_{j'k'} \right) && (2.12) \\ &= \frac{m_{jk}}{p_{jk}} + \lambda_k && (2.13) \end{aligned}$$

where m_{jk} is the number of times that symbol t_j occurs in context $\mathcal{C}_k$. Rearranging, and enforcing the normalisation constraints,

$$\begin{aligned} p_{jk} &= -\frac{m_{jk}}{\lambda_k} && (2.14) \\ &= \frac{m_{jk}}{\sum_{k'} m_{jk'}} && (2.15) \end{aligned}$$

The optimisation can either be performed on a held out set of training data, in which case the counts, m_{jk}, are simply those in the held out set, or after each token in the string being predicted has been observed, resulting in an adaptive model with counts which are updated at each time step.

Although simple, this approach has a number of serious problems. Firstly, even with relatively large training texts, there will be contexts which only appear a small number of times. The maximum likelihood predictions in these cases are unreliable, or in the case of contexts which have never been seen before, ill-defined. More significantly, the maximum likelihood estimate will assign zero probability to symbols which have not been seen in the corresponding context. In most applications a prediction of zero probability is fatal — in arithmetic coding this will result in an infinite length codeword for example. The situation can be improved to some extent by reducing the length of the context. Since the number of

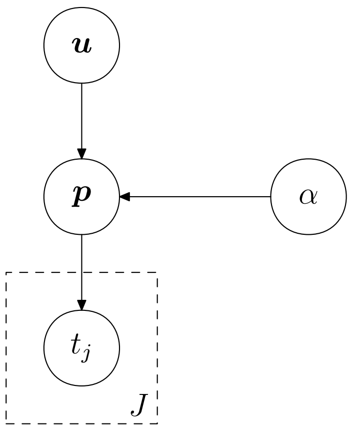


Figure 2.1: A Bayesian network representing a simple language model. Data observed in different contexts are modelled as being independent, with the distribution shown here representing the model for a particular context, $\mathcal{C}$. Each node in the graph represents a variable in the model, with the directed edges representing conditional dependencies. The overall probability distribution can be written as a product of conditional probability tables, where the table for a given variable is dependent on the values of its parents in the graph.

possible contexts grows exponentially in the context length, the number of times that a typical context has been seen is likely to grow rapidly as the context length is shortened (although the situation is complicated somewhat by the non-uniform distribution over tokens in most applications). However this is at the expense of ignoring potentially useful information, and there are still no guarantees that even short contexts occurring at test time will have been observed in the training text. In general, using maximum likelihood estimates directly is not advisable.

2.3.2 Simple Bayesian approaches

A Bayesian approach to the problem treats $P(t \mid \mathcal{C})$, as an unobserved random variable as part of a generative model of the observed data. Let $\boldsymbol{p}_{\mathcal{C}}$ be the vector parameterising this distribution. The posterior distribution, $\Pr(\boldsymbol{p}_{\mathcal{C}} \mid t_1, \ldots, t_M)$ represents a belief in the true value of $\boldsymbol{p}_{\mathcal{C}}$ having observed the string seen so far. As more data is seen, the posterior distribution is updated to represent the newly available information. The width of the posterior becomes smaller, reflecting a growing certainty as to the true value of $\boldsymbol{p}_{\mathcal{C}}$.

More formally, in the generative process $\boldsymbol{p}_{\mathcal{C}}$ is drawn once for the whole data set, and the observed symbols are considered to be independently identically drawn (i.i.d.) from the chosen distribution. This can be represented using a Bayesian network [38] as shown in Figure 2.1 (the meaning of α and $\boldsymbol{u}$ as used in the figure will be explained below). The joint distribution under the model is given by

$$\Pr(\{\boldsymbol{p}_{\mathcal{C}}\}, t_1, \ldots, t_M) = \Pr(\{\boldsymbol{p}_{\mathcal{C}}\}) \prod_{i=1}^{M} \Pr(t_i \mid \mathcal{C}_i, \{\boldsymbol{p}_{\mathcal{C}}\}) \tag{2.16}$$

where $\{\boldsymbol{p}_{\mathcal{C}}\}$ is used to represent the vectors for all context collectively and $\Pr(\{\boldsymbol{p}_{\mathcal{C}}\})$ is a *prior* distribution on $\{\boldsymbol{p}_{\mathcal{C}}\}$. In the following derivation, all probabilities over tokens should be assumed to be conditioned on the appropriate context, which is omitted for clarity. From (2.16), the marginal distribution over the observed data is obtained by integrating over the parameters,

$$\Pr(t_1, \ldots, t_M) = \int \mathrm{d}\{\boldsymbol{p}_{\mathcal{C}}\} \Pr(\{\boldsymbol{p}_{\mathcal{C}}\}) \prod_{i=1}^{M} \Pr(t_i \mid \{\boldsymbol{p}_{\mathcal{C}}\}) \tag{2.17}$$

If $(t_1, \ldots, t_M)$ are observed, the predictive distribution for t_{M+1} is simply

$$\begin{aligned}
\Pr(t_{M+1} \mid t_1, \ldots, t_M) &= \frac{\Pr(t_1, \ldots, t_{M+1})}{\Pr(t_1, \ldots, t_M)} && (2.18)\\
&= \int \mathrm{d}\{\boldsymbol{p}_{\mathcal{C}}\} \frac{\Pr(\{\boldsymbol{p}_{\mathcal{C}}\}) \prod_i \Pr(t_i \mid \{\boldsymbol{p}_{\mathcal{C}}\})}{\Pr(t_1, \ldots, t_M)} \Pr(t_{M+1} \mid \{\boldsymbol{p}_{\mathcal{C}}\}) && (2.19)\\
&= \int \mathrm{d}\{\boldsymbol{p}_{\mathcal{C}}\} \Pr(\{\boldsymbol{p}_{\mathcal{C}}\} \mid t_1, \ldots, t_M) \cdot \Pr(t_{M+1} \mid \{\boldsymbol{p}_{\mathcal{C}}\}) && (2.20)
\end{aligned}$$

Where $\Pr(\{\boldsymbol{p}_{\mathcal{C}}\} \mid t_1, \ldots, t_M)$ is the *posterior distribution* representing the current belief about $\{\boldsymbol{p}_{\mathcal{C}}\}$ given the data seen so far. As with any Bayesian technique, a prior over $\{\boldsymbol{p}_{\mathcal{C}}\}$ is required. A popular choice in many cases is the *Dirichlet distribution* [56]. The Dirichlet distribution is a distribution on the $(k-1)$-simplex, i.e. k-dimensional vectors $\boldsymbol{p}$ such that $\sum_i p_i = 1$ and $p_i \geq 0 \ \forall i$. The distribution is parameterised by a k-dimensional vector $\alpha\boldsymbol{u}$, which is decomposed so that $\boldsymbol{u}$ is a normalised probability vector and α is a positive scalar. The probability density over the simplex is then

$$\Pr(\boldsymbol{p} \mid \alpha\boldsymbol{u}) = \frac{1}{Z(\alpha\boldsymbol{u})} \prod_{i=1}^{K} p_i^{\alpha u_i - 1} \tag{2.21}$$

with normalisation constant

$$Z(\alpha\boldsymbol{u}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha u_i)}{\Gamma(\alpha)} \tag{2.22}$$

Figure 2.2 shows samples from a three dimensional Dirichlet distribution. The mean of the distribution is given by $\boldsymbol{u}$ (also known as the *base measure*), and α is a concentration parameter. For $\alpha < K$ samples will tend to lie in the corners of the simplex, whereas for $\alpha > K$ they are clustered around the mean. As α increases the distribution becomes more peaked. The special case $\alpha\boldsymbol{u} = (1, 1, \ldots)$ corresponds to a uniform distribution over the simplex.

In the absence of any additional information it makes sense to assume a symmetric prior, with a uniform mean, corresponding to $\boldsymbol{u} = (1/|\mathcal{T}|, 1/|\mathcal{T}|, \ldots)$. In this case, the predictive distribution is

$$P(t \mid \mathcal{C}) = \frac{m(t \mid \mathcal{C}) + \alpha u(t)}{M(\mathcal{C}) + \alpha} \tag{2.23}$$

This model is know as Lidstone's law of succession [52]. Particular cases of this include $\alpha =$

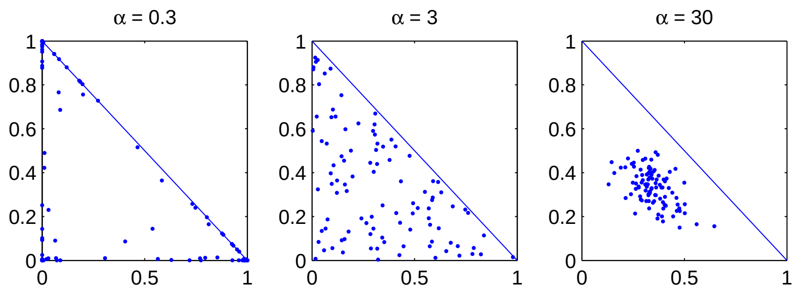


Figure 2.2: Samples from a three dimensional symmetric Dirichlet distribution. From left to right plots are shown for $\alpha = 0.3$, $\alpha = 3$ and $\alpha = 30$. The axes show the first two components of the sampled probability vectors, with the third being constrained by the requirement that they sum to one.

$|\mathcal{T}|$, corresponding to an uninformative prior, in which case Laplace's rule [50] is recovered,

$$P(t \mid \mathcal{C}) = \frac{m(t \mid \mathcal{C}) + \mathbf{1}}{M(\mathcal{C}) + |\mathcal{T}|} \tag{2.24}$$

and $\alpha = |\mathcal{T}|/2$, giving rise to the Jeffreys-Perks law [35]. In the limit $\alpha \to 0$ predictions tend towards the maximum likelihood estimate, although the case $\alpha = 0$ itself isn't a valid Dirichlet distribution.

For any choice of the base measure, many of the problems with the maximum likelihood estimate are avoided; a non-zero probability mass is always assigned to all symbols. If a symmetric base measure is used, a uniform distribution is returned in the absence of observations of a context, which is intuitively appropriate behaviour. Finally, as more and more data is observed the returned distribution tends towards the maximum likelihood estimate. A useful interpretation of α is the number of tokens which need to be seen before previous observations start to play a significant role in the predictive distribution.

2.3.3 Smoothing

Although the Bayesian approach eliminates many of the problems associated with simple maximum likelihood estimates, it still does not make optimal use of the information available from the training data. One piece of information which is potentially useful is the observation that certain contexts tend to result in similar predictions. In particular, contexts which differ by one symbol at the most distant extreme are likely to produce very similar distributions. For example, 'rin' and 'kin' are both contexts where predicting 'g' with high probability would be appropriate. It is therefore likely to be a good idea to share knowledge between these contexts, which may have each only been seen a relatively small number of times. Conversely, unrelated contexts, such as 'rin' and 'fro' are unlikely to result in similar predictions. This idea can be represented by a hierarchy of contexts (see Figure 2.3), with the children of each node being generated by adding symbols at the beginning of the context. This hierarchy is

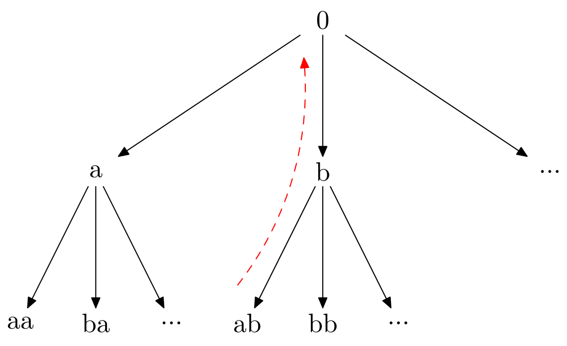


Figure 2.3: The hierarchy of contexts used in smoothing. The red dashed arrow indicates the escape path from ‘ab’ up to the root.

used to share information between nodes which have a common ancestor, with more directly related nodes making more similar predictions.

This process is known as *smoothing* and is often performed using a linear mixture of predictions using different context lengths,

$$P_n\left(t \mid \mathcal{C}\right) = \lambda_n\left(\mathcal{C}\right)\hat{p}_n\left(t \mid \mathcal{C}\right) + \lambda_{n-1}\left(\mathcal{C}\right)\hat{p}_{n-1}\left(t \mid \mathcal{C}\right) + \ldots + \lambda_0\left(\mathcal{C}\right)\hat{p}_0\left(t \mid \mathcal{C}\right) \tag{2.25}$$

where $\hat{p}_i\left(\cdot\right)$ are individual distributions making predictions only using a context of length j, the details of which vary from one model to another. The mixing coefficients, $\{\lambda_i\}$ are subject to the constraint that $\sum_i \lambda_i\left(\mathcal{C}\right) = 1$ so that the mixture is a normalised probability distribution. The 0^{th} order model, $\hat{p}_0$, typically corresponds to a uniform distribution over all terms which amongst other things has the effect of ensuring that a probability of zero is never returned even if all of the finite order components do so individually. The contexts which are smoothed together follow a path from the leaf of the tree shown in Figure 2.3 to the root, with each context being constructed from the previous by deletion of the most distant symbol, for example the context ‘ab’ will be smoothed with predictions in the context ‘b’ as well as the 0^{th} order model, as indicated by the dashed arrow in the diagram. The process of traversing the hierarchy from leaf to root in this fashion is known as *escaping*.

Note that the mixture weights will in general depend on the context and the data seen so far. The choice of these weights, together with the form of the predictions at each order is what differentiates the various approaches which have been used in the past.

A popular method of representing the smoothing process is through *escape symbols*. At each context length, the predictive distributions assign some probability to an additional symbol representing the probability space associated with predictions being made at a lower level. Writing e_n for the probability of the escape symbol as estimated by the n^{th} order model,

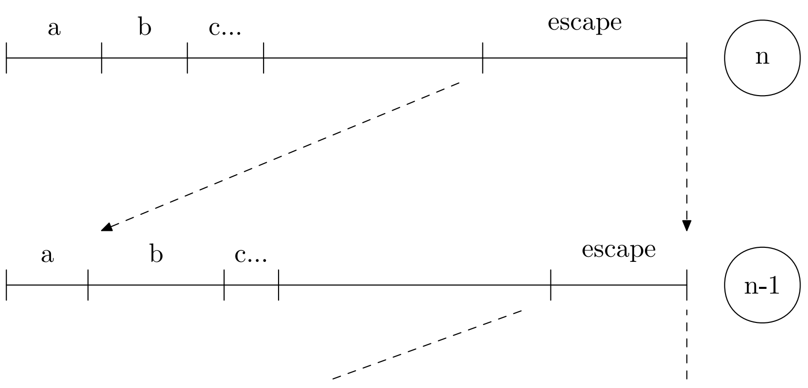


Figure 2.4: Escaping: The top line represents the full code space, and is divided according to the n^{th} level distribution, including some space allocated to the escape symbol. This space, which is represented by the second line is further divided according to the $(n-1)^{th}$ distribution, and in turn contains some space for an escape symbol. The process iterates until the final distribution, which often assigns uniform probability to all symbols is reached. As this distribution does not allocate space to the escape symbol, all of the coding space is now allocated between the symbols.

the overall predictive distribution may be written recursively as

$$P_n\left(t \mid \mathcal{C}\right) = \begin{cases} \left(1-e_n\right) \cdot \hat{p}_n\left(t \mid \mathcal{C}\right) + e_n \cdot P_{n-1}\left(t \mid \mathcal{C}\right) & n > 0 \\ 1/\left|\mathcal{T}\right| & n = 0 \end{cases} \tag{2.26}$$

The process of escaping is illustrated in Figure 2.4. The two formulations are equivalent, with the relationship between e and λ being

$$\lambda_i = \left(1-e_i\right) \prod_{j=i+1}^{n} e_j \tag{2.27}$$

however, it is often the case that the escape symbol representation offers greater conceptual clarity.

In order to implement these models, it is generally necessary to maintain counts for each context in the hierarchy. This is typically achieved using a *trie* data structure [6]. Memory is only allocated in the trie for contexts which have been observed at least once, giving a relatively small storage overhead, and vine pointers can be used to efficiently shorten contexts to the right during the process of escaping.

Interpolation and backing off

There are a number of variations to this process. The approach described above is known as *interpolation* (sometimes referred to as *full blending*). As an alternative, *backing off* (sometimes referred to as *exclusion*) may be used. In this case, predictions are only made at the $(n-1)^{th}$ order level for symbols which have not been assigned any probability mass at the

N-gram	Old Count	New Count
example	0	1
xample	0	1
ample	0	1
mple	5	6
ple	23	23
le	94	94
e	147	147

Table 2.1: An illustration of update exclusion. The centre column contains the counts for each of the N-grams before updating to represent observation of the symbol 'e' in context 'exampl'. The right hand column contains the counts after the update has occurred. The shortest N-gram whose count is incremented is 'mple', which is the longest to have previously not had a non-zero count.

previous level. As the resulting probabilities no longer sum to one, the results are renormalised, with the renormalisation taking place separately for each order, so that the mixing coefficients, λ, remain unchanged. Typically this results in a slight reduction in the performance of the algorithm, but is justified in terms of reducing the computational requirements. A slight technical issue is that the n^{th} order model does not make use of the existing exclusions when making predictions concerning the escape symbol. This means that it will allocate some space to the escape symbol even if all symbols have already been predicted, meaning that there are no predictions to be made at lower orders. This problem can be avoided by using *ad hoc* methods to reduce the probability of the escape symbol in these cases, and becomes less of a problem as more data is observed and the probability of escaping becomes lower.

A further simplification is known as *lazy exclusion*, and works in the same way except that renormalisation is not performed. In this case some quantity of probability mass is never assigned, which is clearly wasteful. For example, if such a model were to be used in an arithmetic compressor there would be gaps in the utilised code space. The advantage is that the computation becomes very simple. The literature reports that lazy exclusion will typically reduce performance by about 5%, as measured by the information rate [6].

Both of these techniques are largely justified on computational grounds, and therefore have become less relevant in the 15 years since they were first developed, as computing power, even in embedded devices, has become much greater.

Update exclusion

In the simplest form of smoothing, the counts used at each order are exactly the number of times that each symbol has previously been seen in the corresponding context. An alternative to this approach is to use *update exclusion*. In this case, when a new symbol is seen, counts are only updated down to the context length one shorter that the shortest context in which the symbol has not been previously seen. An example of this process is shown in Table 2.1.

The effect of update exclusion is that rather than reflecting the number of times that a given N-gram has been seen, the counts for all N-grams shorter than the maximum order reflects the number of extensions to the left of that N-gram which have been seen. For example, suppose that the tri-grams 'aaa', 'baa' and 'caa' had been observed, each occurring more than once, but no other trigrams whose suffix is 'aa' had been seen. In this case, the count associated with the bi-gram 'aa' will be three, rather than the total number of observations of each of the tri-grams. Typically an improvement of around 2% when update exclusion is used, again measured in terms of the information rate, has been reported in the literature [6].

As an illustration of the reasoning behind update exclusion [18] [55], consider pairs of symbols which are often seen together. For example, when modelling at the word level, it may be the case that the word 'United' is often followed by the word 'States'. If update exclusion were not used then the counts for 'United' and 'States' following the zero-length context would be approximately equal. Suppose a new context is seen. As no data exists for this context, predictions must be entirely based on the uni-gram probabilities and lower, and therefore the two words would be predicted with almost equal probability. If update exclusion is used then the word 'States' would only have its count incremented the first time that the phrase 'United States' is observed. After many observations of the phrase, predictions in a novel context would assign significantly more probability to 'United' as would be expected.

2.3.4 Linear interpolation

One simple approach to smoothing is to simply to treat the mixture coefficients, λ in (2.25) as parameters to be learned, together with the component distributions themselves. This approach is known as Jelinek–Mercer smoothing [37]. The values of λ can be estimated using held-out data separate from that used to train the distributions, or can be estimated at the same time using an approach known as *deleted interpolation*. In general, a separate λ can be introduced for each context, although this will not be practical as there is unlikely to be sufficient data to obtain reliable predictions. Furthermore, forcing the mixture coefficients to be independent runs against the idea of smoothing, which is to encourage sharing of information between related contexts. Instead, contexts are assigned to 'buckets', with all members of the same bucket getting the same value. Various schemes exist for 'bucketing' the contexts, for example, see [37], [2] and [17].

2.3.5 PPM and Witten–Bell smoothing

Prediction by Partial Match (PPM) is a family of compression algorithms making use of arithmetic coding and language modelling [21] [6] [99]. In its earliest form, PPM-A, the language modelling component introduces an extra count for each context which is associated with the escape symbol. Expressing this in terms of the notation used in (2.26), symbols are predicted with probability

$$\hat{p}_n\left(t \mid \mathcal{C}\right) = \frac{m_n\left(t \mid \mathcal{C}\right)}{M_n\left(\mathcal{C}\right)} \tag{2.28}$$

and the escape symbol with probability

$$e_n = \frac{1}{M_n(\mathcal{C}) + 1} \tag{2.29}$$

This approach can be generalised by replacing the single additional count with a parameter α_n, with $\alpha_n > 0$. In general, this parameter is allowed to depend on the order at which predictions are being made. This substitution results in a new escape probability of

$$e_n = \frac{\alpha_n}{M_n(\mathcal{C}) + \alpha_n} \tag{2.30}$$

We refer to the resulting model as *generalised PPM-A*.

A variation on this scheme is to make α_n dependent on the context, for example by setting α_n equal to the number of *different* symbols which have previously been seen in the current context. This has the intuitively appealing effect of causing more variation in the predictions in contexts which have already been observed to be followed by a wide variety of different symbols, and is the approach used in PPM-C [60], otherwise known as Witten–Bell smoothing. Many other members of the PPM family exist, of which PPM-B and PPM-D will be examined in relation to Kneser–Ney smoothing later.

Generalised PPM-A is the basis for much of the work in the remainder of this Chapter, and will be re-examined in much greater detail in later sections.

2.3.6 Katz and Church–Gale smoothing

The Good–Turing estimate is an alternative way of estimating a probability distribution from observed samples [62]. Unlike Laplace's law of succession, the Good–Turing estimate is intended for the case when the total number of symbols is not known in advance. It is therefore inappropriate for symbol-level models which make use of a known finite alphabet, but can be used when the symbols are words. The method involves discounting the counts for observed tokens such that the new count is given by

$$r^\star = (r+1)\frac{E(M_{r+1})}{E(M_r)} \tag{2.31}$$

where r is the unadjusted count for the token in question and $E(M_i)$ is an estimate of the number of tokens which have been observed precisely i times previously. There are a number of ways of estimating $E(M_i)$, which are to some extent motivated by the need to avoid pathological cases such as will occur if $E(M_i) = 0$ for any i. Having performed discounting, the predictive distribution is then defined by dividing through by the *original* total number of observations. In general the effect of discounting will be to leave some unused probability space which is interpreted as being the probability that the next observation will be a previously unseen token.

Katz smoothing [40] is based on the Good–Turing estimate, but only discounts the counts for tokens which have been observed fewer times than some threshold (the value 5 is suggested in Katz's paper). In order to preserve certain desirable properties of the distribution, the Good–Turing estimate is modified to compensate for the distortion introduced by the threshold.

A more complicated approach is used in Church–Gale smoothing [19]. This scheme is best defined in the bi-gram case, where bi-grams are binned according to the product of the counts of the two constituent terms. The Good–Turing estimate is then used separately within each of the bins.

These algorithms are unfortunately not based on a clear theoretical foundation, so while they do give good performance in some cases, it is difficult to perform a more in-depth analysis. They will therefore not be considered further in this thesis.

2.3.7 Absolute discounting and Kneser–Ney smoothing

Both Absolute discounting and Kneser–Ney smoothing introduce a constant, β, which is subtracted from the raw counts to get new values. Normalisation is still performed using the full counts, resulting in a certain amount of left-over space which is used for the escape symbol. Recasting this in terms of escape probabilities gives

$$\hat{p}_n (t \mid \mathcal{C}) = \frac{\max (m_n (t \mid \mathcal{C}) - \beta, 0)}{M_n (\mathcal{C}) - q_n (\mathcal{C}) \beta} \tag{2.32}$$

$$e_n = \frac{q_n (\mathcal{C}) \beta}{M_n (\mathcal{C})} \tag{2.33}$$

where $q_n (\mathcal{C})$ is the number of distinct symbols which have previously been observed in context $\mathcal{C}$. Kneser and Ney suggest that β be defined according to

$$\beta = \frac{n_1}{n_1 + 2n_2} \tag{2.34}$$

where n_1 and n_2 are the total number of N-grams with counts of 1 and 2 respectively. Absolute discounting [63] refers to the use of this method with raw counts for lower order context, whereas Kneser–Ney smoothing [44] makes use of update exclusion. The same approach is used in PPM-B and PPM-D, which stipulate $\beta = 1.0$ and $\beta = 0.5$ respectively. Chen and Goodman propose a generalisation of Kneser–Ney smoothing in which different values of β are used for symbols which have been seen different numbers of times [18].

2.3.8 Reduced context dimensionality

A number of approaches exist which essentially aim to reduce the dimensionality of the context, so that similar contexts are mapped to the same representation. One way of doing this is to attempt to cluster terms, either using some objective criterion such as part-of-speech, or inferring a clustering based on the statistics of the language. The cluster indices can then either partially or totally replace the terms themselves in the context.

An alternative method is to learn a low-dimensional representation of the context as a whole from data, and use this representation to inform the model as to how information can be shared between contexts. This is the approach taken by the *neural probabilistic language model* [7] [8], which uses a neural network to learn such a representation. A similar approach was taken by Blitzer *et al.* [13], who learn a mapping from contexts to low dimensional vectors.

2.3.9 Maximum entropy language models

Let $\boldsymbol{x}$ be a random variable, and let $f_i(\boldsymbol{x})$ be a set of functions on $\boldsymbol{x}$. Suppose that we wish to define a distribution P over $\boldsymbol{x}$, such that for all i, the constraint

$$\sum_{\boldsymbol{x}} P(\boldsymbol{x}) f_i(\boldsymbol{x}) = K_i \tag{2.35}$$

is satisfied. The maximum entropy principle [34] states that in the absence of additional information it is appropriate to chose the distribution which gives maximal entropy while meeting the constraints. It can be shown that for constraints of the form given in (2.35) the maximum entropy distribution will have the form

$$P(\boldsymbol{x}) = \frac{1}{Z} \prod_i \exp(\lambda_i f_i(\boldsymbol{x})) \tag{2.36}$$

where $\{\lambda_i\}$ are parameters to be determined and Z is a normalising constant necessary to ensure that the probabilities sum to one. Furthermore, if the K_i are the expected values of the features under the empirical distribution of an observed dataset, then the values of $\{\lambda_i\}$ are those which maximise the likelihood of the data.

The maximum entropy approach has been used in language modelling, for example in [78]. One advantage of this approach is the flexibility with which $\{f_i\}$ can be chosen. Unigram statistics can be incorporated using features such as

$$f_t(t_i, \mathcal{C}_i) = \begin{cases} 1 & t_i = t \\ 0 & t_i \neq t \end{cases} \tag{2.37}$$

which represents the statistics of a particular token t (features of this kind are introduced for all members of the token set). Similarly, higher order N-grams can be introduced as binary features which fire only if the token being predicted and the context match a given pattern. Alternate features may take into account, for example longer range correlations, or trigger words, the presence of which anywhere in the context might affect predictions.

Empirically the maximum entropy approach has worked well. However, it is reported to be prone to over-fitting, and it is not clear the extent to which its success depends on heuristics such as early stopping and careful choice of prior distributions.

2.3.10 Topic models

Topic models assume that there is a hidden variable associated with the text representing a 'topic'. One of the most recent approaches is Latent Dirichlet Allocation (LDA) [12], which in turn is based on the earlier approach of Latent Semantic Indexing (LSI) [26] and probabilistic LSI (pLSI) [32]. All of these methods make use of training data which is divided into documents, representing each document as being composed of a number of 'topics'. In pLSI, a distribution over topics is provided for each document in the collection, and a distribution over tokens is provided for each topic. The generative process can be viewed as using the distribution over topics to draw a latent variable, z_n, for each position in the document, and then drawing a token from the distribution associated with that topic. Mathematically speaking,

$$P\left(\{t_n\} \mid d\right) = \prod_n \sum_{z_n} P\left(t_n \mid z_n\right) P\left(z_n \mid d\right) \tag{2.38}$$

where d is an index over documents in a training set. Note that this model does not specify a prior over the per-document topic distribution, $P\left(z_n \mid d\right)$. If a new document is observed then it is not clear how to provide a distribution over topics. This limitation is addressed by LDA, which assigns a Dirichlet prior over the topic distributions for each document. In other words, if

$$P\left(z_n \mid d\right) = \theta_{z_n}^{(d)} \tag{2.39}$$

then LDA requires that

$$\boldsymbol{\theta}^{(d)} \sim \text{Dirichlet}\left(\boldsymbol{\alpha}\right) \tag{2.40}$$

The hyperparameter $\boldsymbol{\alpha}$ is shared between all documents, allowing information to be shared in the form of a 'base' distribution over topics.

Currently, topic models in general do not capture statistical correlations between terms, using only uni-gram statistics. However, they have been successfully used to reveal underlying structure in natural language documents [29]. The relationship between LDA and various other approaches in language modelling and information retrieval will be discussed further in later sections.

2.4 Experimental evaluation of generalised PPM-A

The following sections are devoted to a detailed analysis of generalised PPM-A from a Bayesian viewpoint. Before considering how Bayesian methods can be used to understand generalised PPM-A, this section describes an empirical analysis of the method which will confirm results in the literature and serve as a baseline for use in later sections.

We performed these evaluations using two samples of typical English text, `alice29.txt` from the Canterbury Corpus and a section from the Enron corpus (see Appendix A), with performance being measured using the information rate as defined above. The model was tested non-adaptively; after training the counts were held fixed, so no further learning took

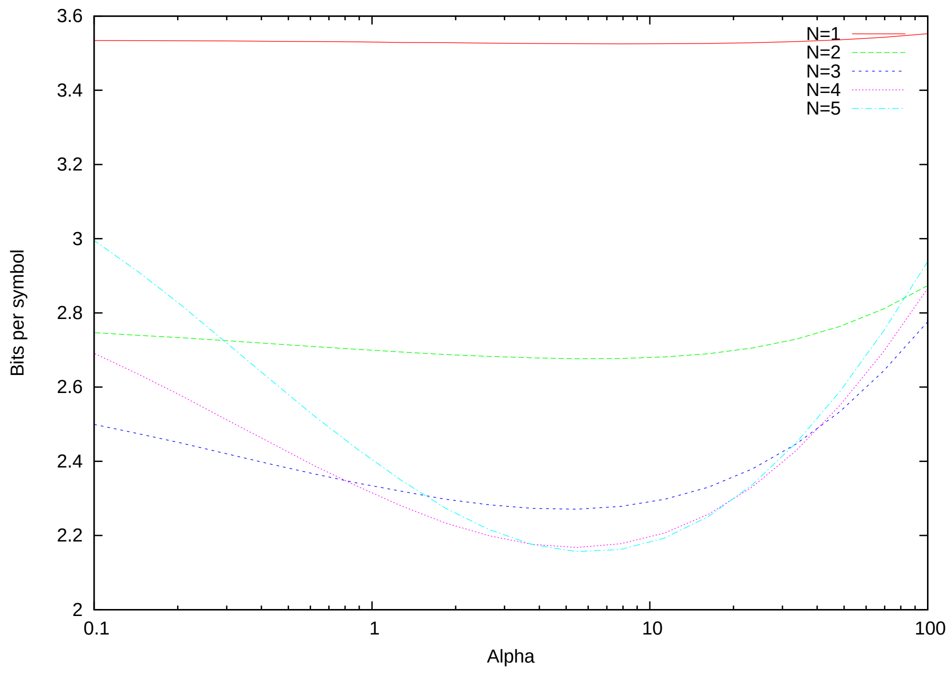


Figure 2.5: Experimental evaluation of the general PPM-A language model for values of N between 1 and 5, using an 100kB extract from `alice29.txt` for training and a separate 10kB extract from the same text for testing. For $N = 2$ the minimum information rate was achieved at $\alpha = 6.50$, giving 2.68 bits per symbol, whereas for $N = 5$ the minimum was at $\alpha = 6.07$, giving 2.16 bits per symbol.

place during the test phase. In all cases update exclusion was used unless stated otherwise, and full interpolation was performed on the predictive distribution. Where values of α are given without subscript the same value is used at all orders.

2.4.1 Baseline performance and varying alpha

In the first experiment we looked at the variation of the information rate as a function of the smoothing parameter, α, and the maximum order, N. The evaluation was performed by training the model on an extract of size 100kB, and then evaluating the performance on a disjoint extract of 10kB. Results of this evaluation are shown in Figures 2.5 and 2.6.

As would be expected, the performance improves as N increases, although there is very little difference between the best performance for $N = 4$ and $N = 5$. In all cases an optimum is found between around $\alpha = 6$ and $\alpha = 6.5$. Note however that as N increases the minimum becomes narrower, and therefore away from the optimum value of α *worse* performance may actually be obtained for larger N. Although the variation of performance with α is similar for the two texts considered here, in general it may well be the case that the optimal value of α is dependent on the text being modelled. It might be expected that this would be more likely to be the case with language models working at the level of whole words, as a result of a higher degree of variability in word level statistics. In this case, any fixed value of α specified

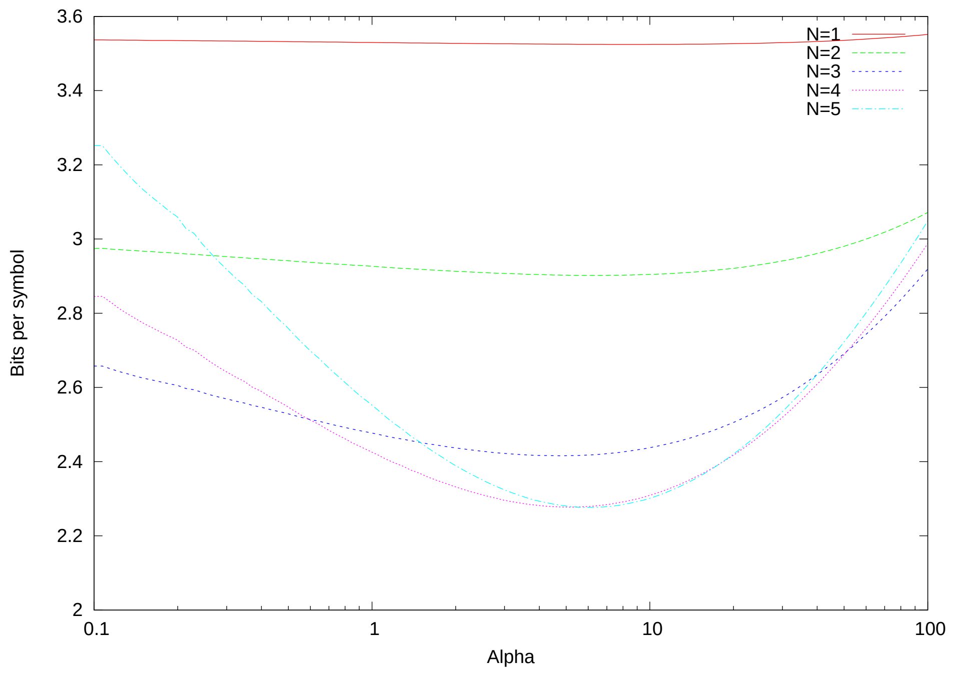


Figure 2.6: Experimental evaluation of the general PPM-A language model for values of N between 1 and 5, using an 100kB extract from the Enron corpus for training and a separate 10kB extract from the same text for testing. For $N = 2$ the minimum information rate was achieved at $\alpha = 6.31$, giving 2.90 bits per symbol, whereas for $N = 5$ the minimum was at $\alpha = 6.06$, giving 2.28 bits per symbol.

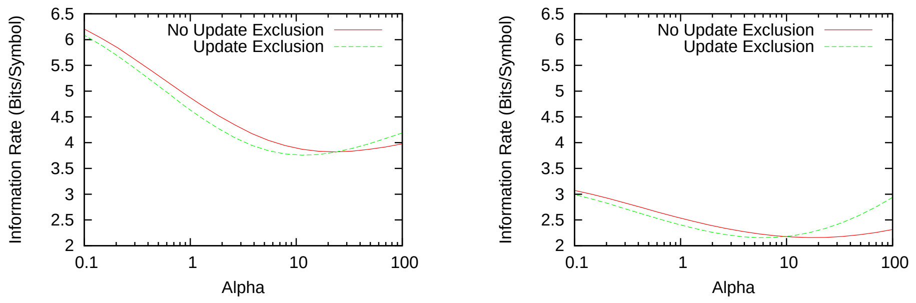


Figure 2.7: Effect of update exclusion. Compression rate as a function of α for training on 1kB and testing on 1kB (left) and for training on 100kB and testing on 10kB (right) using `alice29.txt`.

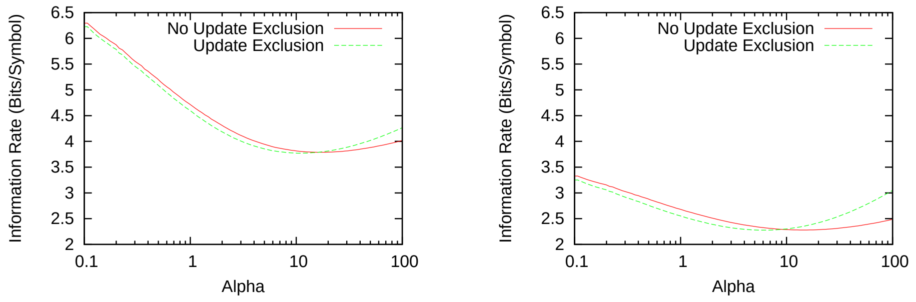


Figure 2.8: Effect of update exclusion. Compression rate as a function of α for training on 1kB and testing on 1kB (left) and for training on 100kB and testing on 10kB (right) using the Enron corpus.

in advance will generally not be optimal. The graphs presented here show that for values of α away from the optimum there is some value of N above which better performance cannot be reliably obtained. This observation potentially helps to explain why attempts to produce models with unbounded context lengths [20] have generally not been successful. The results shown here are broadly comparable to those previously reported in the literature.

2.4.2 Evaluation of update exclusion

We next performed a second evaluation to test the effect of update exclusion. Two cases were tried: firstly a short text was used, 1kB of training text and 1kB of test text. Secondly, a longer text of 100kB for training and 10kB for test was evaluated. The results are shown in Figures 2.7 and 2.8.

The results for the shorter text show a clear advantage to update exclusion, whereas those for the longer text indicate that roughly equal performances are possible in this case. Note that the value of α at which the optimum is attained changes as update exclusion is enabled. For the shorter text, if α is held fixed at 1.0 as in the original PPM-A implementation then update exclusion gives an improvement of 4.9% for `alice29.txt` and 2.5% for the Enron text. However if the optimal values are considered, improvements of 1.7% and 0.47%

respectively are observed. Similarly, for the longer text the improvements at $\alpha = 1$ are 5.2%and 4.8%. Comparing optimal values reveals improvements of 0.10% and 0.13%. Note that the improvements when considering optimal values are greater for the shorter text. This suggests that larger improvements may be observed if evaluation is done on-line, i.e. if predictions are made before each symbol in the training text is observed, as in this case the initial predictions are made with very little training data. The results also indicates that the improvement due to update exclusion will be dependent on the value of α which is chosen. For comparison, a figure of 2% is given in the literature [6] for PPM.

2.5 Hierarchical Bayesian methods

Section 2.3.2 discussed Bayesian methods for estimating predictive distributions based on example data, but did not consider sharing information between contexts. In Section 2.3.3 a number of *ad hoc* methods for sharing information between contexts were discussed. The remainder of this chapter deals with the combination of the two, giving a principled Bayesian method which is able to share information between related contexts. The goal will be to relate this approach to generalised PPM-A.

2.5.1 The hierarchical Dirichlet distribution

The Dirichlet distribution, described in Section 2.3.2, provides a way of assigning a prior distribution over single probability vectors. The hierarchical Dirichlet distribution is a generalisation of the Dirichlet distribution, providing a joint prior over sets of related distributions. This can be used as part of a generative model of grouped discrete data, where each group is i.i.d. from one of the discrete distributions. x_{jk} is used to represent the k^{th} sample in group j. Consider first the two-level model, which is shown as a graphical model in Figure 2.9. In this model, a probability vector, $\boldsymbol{p}$, is first drawn from a Dirichlet distribution with a given parameter, $\alpha_1\boldsymbol{u}$. Having obtained this vector, it is multiplied by a constant, α_2, and is then used as the base measure for a second Dirichlet distribution, from which a vector, $\boldsymbol{q}_j$ is drawn from each group,

$$\boldsymbol{p} \sim \text{Dirichlet}\left(\alpha_1\boldsymbol{u}\right) \tag{2.41}$$

$$\boldsymbol{q}_j \sim \text{Dirichlet}\left(\alpha_2\boldsymbol{p}\right) \tag{2.42}$$

$$x_{jk} \sim \text{Categorical}\left(\boldsymbol{q}_j\right) \tag{2.43}$$

The two scalars, α_2 and α_1 are considered as parameters of the model, specified in advance. Roughly speaking, α_1 describes the concentration of $\boldsymbol{p}$, and therefore the distributions themselves, around the mean $\boldsymbol{u}$. α_2 describes the degree of variation between the distributions for different groups. The use of a shared parameter vector which itself is a random variable introduces a sharing of information between the individual distributions. This is appropriate in cases where the underlying distribution in each group is believed to be similar, but where

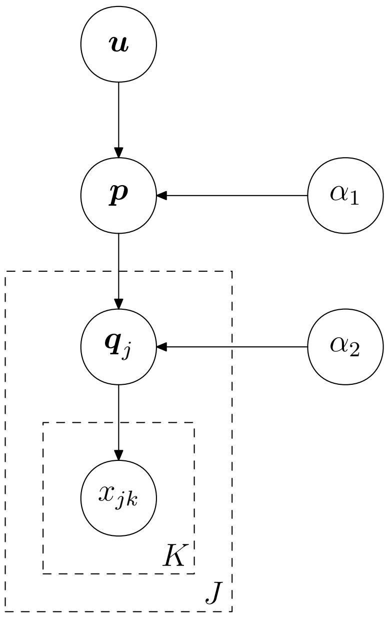


Figure 2.9: Bayesian network representing the hierarchical Dirichlet distribution. $\boldsymbol{u}$ is a base distribution, considered to be a parameter of the model, which is used to generate $\boldsymbol{p}$ from a Dirichlet distribution. This value is in turn used to generate a set of probability vectors, $\{\boldsymbol{q}_i\}$, from which the data is drawn.

some degree of variation is believed to exist. Observations made in one group can affect the posterior distribution of $\boldsymbol{p}$, and therefore alter future predictions for observations in other groups.

2.5.2 Deeper hierarchies

The same idea can be applied to deeper hierarchies in a straightforward way. In this case, the diagram shown in Figure 2.9 is simply extended to represent a deeper tree of probability vectors. The vector for each node is drawn from a Dirichlet distribution parameterised by the distribution corresponding to the parent node multiplied by a constant, α_i. Additional α parameters are introduced as required.

To give a concrete example, consider the extension to a three level model, where x_{jkl} represents the l^{th} sample from group k within parent group j. In this case, the distribution becomes

$$\begin{aligned}
\boldsymbol{p} &\sim \text{Dirichlet}\left(\alpha_1 \boldsymbol{u}\right) && (2.44)\\
\boldsymbol{q}_j &\sim \text{Dirichlet}\left(\alpha_2 \boldsymbol{p}\right) && (2.45)\\
\boldsymbol{q}_{jk} &\sim \text{Dirichlet}\left(\alpha_3 \boldsymbol{q}_j\right) && (2.46)\\
x_{jkl} &\sim \text{Categorical}\left(\boldsymbol{q}_{jk}\right) && (2.47)
\end{aligned}$$

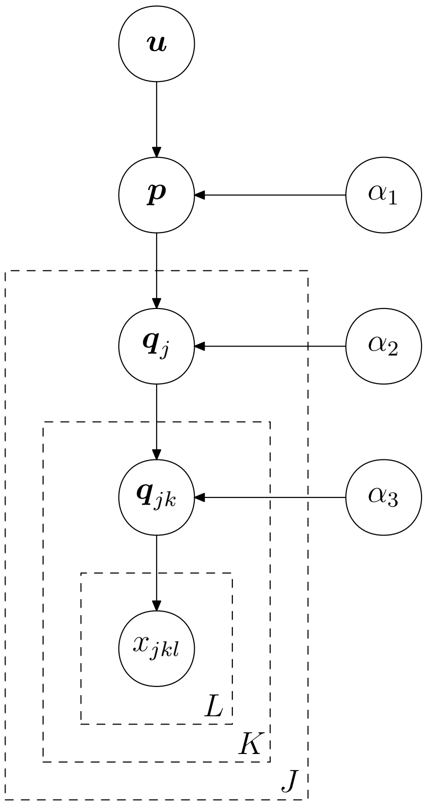


Figure 2.10: Bayesian network representing the three level hierarchical Dirichlet distribution.

where $\boldsymbol{q}_{jk}$ have been introduced as the distribution for each group, with the groups now being clustered according to the value of j. The three level Bayesian network is shown in Figure 2.10.

The use of a deeper hierarchy means that the correlations between distributions reflect a tree structure over the groups. Consider two distributions, $\boldsymbol{q}_{j_1k_1}$ and $\boldsymbol{q}_{j_2k_2}$. If $j_1 = j_2$ then both distributions will have been drawn from the same Dirichlet, with parameter $\alpha_2\boldsymbol{q}_{j_1}$. However, if $j_1 \neq j_2$ then the relationship will be more distant, related only by the fact that their parent distributions will have been drawn from the same Dirichlet, with parameter $\alpha_1\boldsymbol{p}$.

2.5.3 The hierarchical Dirichlet language model

The hierarchical Dirichlet language model [55] is an application of the hierarchical Dirichlet distribution to language modelling. The approach is a generalisation of that described in Section 2.3.2, with a separate probability vector being used for terms in each context. In order to share information between contexts, a single hierarchical Dirichlet prior is used for all of the distributions. The structure of the model directly mirrors the hierarchy of contexts described in Section 2.3.3.

The original implementation of the hierarchical Dirichlet language model used an approximation in order to infer $\boldsymbol{p}$ directly in a bi-gram model. The model was compared to deleted interpolation (see Section 2.3.4), and was found to give comparable performance, with the difference in performance being dependent on the number of λ values used in deleted interpolation.

The use of a shared Dirichlet prior between contexts in the hierarchical Dirichlet language model can be viewed as playing a similar role to that shared between documents in latent Dirichlet allocation as described in Section 2.3.10. While in the latter case the shared prior allows information to be shared between documents concerning the distribution over topics, in the hierarchical Dirichlet language model the shared prior allows information to be shared concerning the distribution over tokens in different contexts.

2.5.4 Pólya urns and oracles

Suppose that a probability vector $\boldsymbol{p}$ is randomly drawn from a Dirichlet distribution with parameter $\alpha\boldsymbol{u}$, and that M values, $\{x_1, \ldots, x_M\}$ are drawn from $\boldsymbol{p}$. Let m_i be the number of times that value i appears in the samples. The probability of the observed data is given by marginalising over all values of $\boldsymbol{p}$,

$$\begin{aligned}
\Pr(x_1, \ldots, x_M) &= \int \mathrm{d}\boldsymbol{p} \Pr(\boldsymbol{p}, x_1, \ldots, x_M) && (2.48)\\
&= \int \mathrm{d}\boldsymbol{p} \Pr(x_1, \ldots, x_M \mid \boldsymbol{p}) \Pr(\boldsymbol{p}) && (2.49)\\
&= \frac{\prod_i \Gamma(\alpha u_i)}{\Gamma(\alpha)} \int \mathrm{d}\boldsymbol{p} \prod_i p_i^{m_i} \prod_i p_i^{\alpha u_i - 1} && (2.50)\\
&= \frac{\Gamma(\alpha)}{\Gamma(\alpha + M)} \prod_i \frac{\Gamma(\alpha u_i + m_i)}{\Gamma(\alpha u_i)} && (2.51)\\
&= \frac{\prod_i (\alpha u_i)^{[m_i]}}{\alpha^{[M]}} && (2.52)
\end{aligned}$$

in which the last line makes use of the *rising factorial*,

$$x^{[n]} = x \cdot (x+1) \cdots (x+n-1) \qquad (2.53)$$

Suppose that an additional data point is observed, taking value j. The probability of the new data set has the same form, but with one additional count. In other words,

$$\Pr(x_1, \ldots, x_{M+1}) = \frac{\prod_i (\alpha u_i)^{[m_i + \delta_{ij}]}}{\alpha^{[M+1]}} \qquad (2.54)$$

The conditional probability can therefore be found by taking the ratio of these,

$$\Pr\left(x_{M+1}=j \mid x_1,\ldots,x_M\right) = \frac{\Pr\left(x_1,\ldots,x_{M+1}\right)}{\Pr\left(x_1,\ldots,x_M\right)} \tag{2.55}$$

$$= \frac{\alpha u_j + m_j}{\alpha + M} \tag{2.56}$$

$$= \frac{M}{\alpha+M}\cdot\frac{m_j}{M} + \frac{\alpha}{\alpha+M}\cdot u_j \tag{2.57}$$

In the last line, the predictive distribution has been separated into two terms to clarify its interpretation. With probability proportional to M, the next datum is drawn based on the previous observations, with the value being proportional to the number of times it has been seen before (equivalent to the maximum likelihood estimate). With probability proportional to α, a new sample is drawn from the base distribution, $\boldsymbol{u}$, instead. A metaphor which will be useful later on is to consider the latter case as asking an *oracle* for a prediction. In this case the oracle will always return draws from a fixed distribution, $\boldsymbol{u}$, but the behaviour of the oracle will be generalised in later sections. Local counts are updated after each sample regardless of how it is generated.

An alternative metaphor is to consider an urn initially containing αu_i balls of a particular colour for each i. Balls are repeatedly drawn from the urn and their colour noted. After each ball has been drawn it is replaced, and another ball of the same colour is added. It is this analogy which gives rise to the name of the sampling scheme. This method provides an alternative to following the generative model directly. If samples are drawn repeatedly from the predictive distribution, with the distribution itself being updated to reflect the new data, the resulting data will have the same distribution as if a value for $\boldsymbol{p}$ was drawn directly and the data independently from that.

2.5.5 Sampling from the hierarchical Dirichlet distribution

The Pólya urn sampling scheme can be generalised to the hierarchical case by noting that the nature of the prior is only important when a new sample is drawn, corresponding to the second term in (2.57). In other words, in the hierarchical case it is simply a matter of replacing the fixed distribution used by the oracle with another Dirichlet distribution. In this case, when asked, the oracle itself allocates some probability to asking another, top-level oracle, which in the two level case draws from a fixed distribution. As multiple groups use the same distribution, $\boldsymbol{p}$ in their base measure, the same mid-level oracle is shared between all of the groups. Let $\bar{m}_n\left(t \mid \mathcal{C}\right)$ be the counts used when making predictions by the n^{th} level oracle corresponding to context $\mathcal{C}$, and $\bar{M}_n\left(\mathcal{C}\right)$ be the total number of counts for that oracle. For notational consistency the same symbol will be used for the raw counts at the bottom level, even though this does not strictly speaking represent an oracle. The counts used by the mid-level oracle are the number of times that it has been asked *by any group*, and has returned the corresponding symbol. It is this mechanism which allows information to be shared between the groups. The oracle mechanism is illustrated graphically in Figure 2.11.

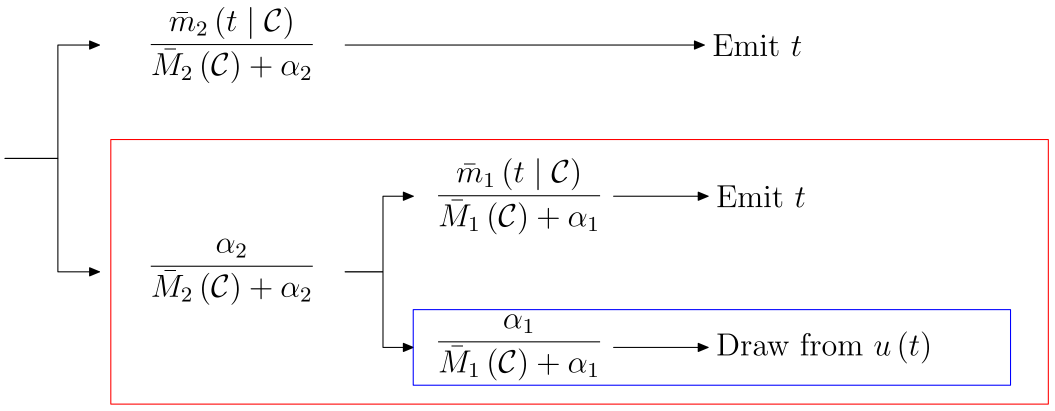

$\frac{\bar{m}_2(t \mid \mathcal{C})}{\bar{M}_2(\mathcal{C}) + \alpha_2}$
Emit t
$\frac{\alpha_2}{\bar{M}_2(\mathcal{C}) + \alpha_2}$
$\frac{\bar{m}_1(t \mid \mathcal{C})}{\bar{M}_1(\mathcal{C}) + \alpha_1}$
Emit t
$\frac{\alpha_1}{\bar{M}_1(\mathcal{C}) + \alpha_1}$
Draw from $u(t)$

Figure 2.11: A schematic representation of the hierarchical Dirichlet distribution viewed in terms of a oracles. A red box is used to outline the mid level oracle, and a blue box indicates the top-level oracle, which in this case always draws from the distribution $u(t)$.

For distributions involving more complicated tree structures, it is simply a matter of adding additional oracles representing the extra nodes present in the tree. This representation is the finite equivalent of the oracle used in the infinite hidden Markov model [4], or the Chinese restaurant franchise [92].

As mentioned above, the counts used by the oracles are dependent on the number of times that the oracle has been asked in the past, as well as the values ultimately returned for the corresponding data. This information is referred to collectively as the *oracle path.* If data is being generated by sequentially sampling using the oracle mechanism this dependency is not a problem — the oracle path is known. However, in many situations this is not the case. Given a set of samples only, the predictive distribution for the next datum involves a marginalisation over all possible oracle paths. The number of such paths grows exponentially with the size of the data set[2], so it rapidly becomes intractable to explicitly perform the marginalisation. It is therefore necessary to use approximations, which will be the subject of the next section

2.6 Bayesian interpretation of generalised PPM-A

Consider the predictive distribution under the hierarchical Dirichlet distribution when the *oracle path* is known. In this case, the counts maintained by each oracle will be known, and predictions can therefore be made according to the mechanism described in Section 2.5.5. For

[2]In the two-level case, imagine a variable associated with each datum indicating whether it was generated by asking the oracle or not. These variable are essentially unconstrained, with the exception of the first time each value appears in each group which must be generated by the oracle.

the two level case, the predictive distribution has the form

$$P_2(t \mid \mathcal{C}) = \underbrace{\frac{\bar{M}_2(\mathcal{C})}{\bar{M}_2(\mathcal{C}) + \alpha_2} \cdot \frac{\bar{m}_2(t \mid \mathcal{C})}{\bar{M}_2(\mathcal{C})}}_{\text{Local Prediction}} + \frac{\alpha_2}{\bar{M}_2(\mathcal{C}) + \alpha_2} \left(\underbrace{\frac{\bar{M}_1(\mathcal{C})}{\bar{M}_1(\mathcal{C}) + \alpha_1} \cdot \frac{\bar{m}_1(t \mid \mathcal{C})}{\bar{M}_1(\mathcal{C})}}_{\text{Mid-level}} + \underbrace{\frac{\alpha_1}{\bar{M}_1(\mathcal{C}) + \alpha_1} u(t)}_{\text{Top-level}} \right) \tag{2.58}$$

For a general n-level hierarchy, this expression can be written recursively as

$$P_n(t \mid \mathcal{C}) = \begin{cases} \frac{\bar{M}_n(\mathcal{C})}{\bar{M}_n(\mathcal{C}) + \alpha_n} \frac{\bar{m}_n(t \mid \mathcal{C})}{\bar{M}_n(\mathcal{C})} + \frac{\alpha_n}{\bar{M}_n(\mathcal{C}) + \alpha_n} P_{n-1}(t \mid \mathcal{C}) & n > 0 \\ u(t) & n = 0 \end{cases} \tag{2.59}$$

The predictive distribution for generalised PPM-A can be written in a similar way. Substituting (2.28) and (2.30) into (2.26) gives

$$P_n(t \mid \mathcal{C}) = \begin{cases} \frac{M_n(\mathcal{C})}{M_n(\mathcal{C}) + \alpha_n} \frac{m_n(t \mid \mathcal{C})}{M_n(\mathcal{C})} + \frac{\alpha_n}{M_n(\mathcal{C}) + \alpha_n} P_{n-1}(t \mid \mathcal{C}) & n > 0 \\ 1/|\mathcal{T}| & n = 0 \end{cases} \tag{2.60}$$

Comparison of (2.59) and (2.60), assuming that $u(t)$ is a uniform distribution over the token set, reveals an equivalence between the predictive distributions for the hierarchical Dirichlet language model and generalised PPM-A. The oracle counts in the hierarchical Dirichlet model play a role equivalent to the counts used at the n^{th} order in PPM-A.

As mentioned above, the oracle path, and therefore the counts used to make the predictions are not generally know, however, it is possible to make an approximation by assuming a specific oracle path when making predictions. Two specific paths are considered here:

- **Minimal Oracle Assumption:** Predictions are based on the assumption that the oracle was only asked when there was no other choice, i.e. the first time that a symbol was seen in the associated context. Referring back to Section 2.3.3, the counts used by the oracle in this case are exactly equivalent to the counts which are obtained at each level when update exclusion is used.

- **Maximal Oracle Assumption:** Predictions are based on the assumption that the oracle was asked at every data point, and that the enquiry propagated all the way to the root of the tree. This is equivalent to using raw counts in generalised PPM-A.

To reiterate, there is a direct equivalence between generalised PPM-A and the hierarchical Dirichlet language model under the two approximations given above. These approximations are equivalent to generalised PPM-A with and without update exclusion respectively. Intuitively one would expect that the minimal oracle assumption would be a better approximation for small values of α. Expression (2.59) shows that the probability of making an oracle enquiry (conditioned on a fixed history) is proportional to α and therefore in the limit of small α oracle enquiries will only take place when there is no other option. The remainder of this chapter assesses this intuition experimentally to establish the performance of the two approximations under various conditions.

Before leaving this section, the interpretation presented above can help understand a result presented in Section 2.4.1. In particular that the peak in performance of PPM-A becomes narrower as the context length is increased. Having interpreted PPM-A as an approximation to the hierarchical Dirichlet model, an increase in the context length can be viewed in terms of increasing the number of times that a sample is drawn from a Dirichlet distribution to obtain the predictive distribution for a given context. As α is used in each of these samples, a possible explanation of the narrowing of the peak is that the effect of a mismatch in α is magnified each time a sample is drawn. In the case of a deeper hierarchy the mismatch will therefore be magnified more, resulting in relatively poorer performance.

2.6.1 Backing off and interpolation

The full predictive distribution under either of the assumptions described in the previous section corresponds to full interpolation in the language modelling context. An alternative would be to make predictions under the assumption that escaping will take place to a particular depth. For example, in keeping with the minimal oracle assumption is the possibility that predictions should be made assuming that no escaping will take place unless it is absolutely necessary.

This results in a predictive distribution which is very similar to that obtained when exclusion is used. However, the two are not quite identical — using exclusion predictions made at each context length are renormalised to take into account the symbols which have been excluded *before* mixing takes place, whereas in the hierarchical model renormalisation takes place *after* mixing. In other words, this is equivalent to lazy exclusion with renormalisation afterwards to avoid wasted space.

From a theoretical viewpoint, there is no good reason to perform exclusion, as full blending is closer to the hierarchical model and the use of one approximation does not justify the use of another. Backing off will therefore not be considered further in this thesis.

2.7 Empirical evaluation

The analogy presented in Section 2.6 opens up the possibility of using empirical means to further understand generalised PPM-A. The conditions in which either of the two approximations hold, and in which one is better than the other or vice-versa is of great interest. The following sections address these issues, starting with a Monte Carlo comparison of the two approximations. Later chapters will use Monte Carlo methods to examine the prior distribution provided by the hierarchical Dirichlet model, as well as the posterior distribution under this prior conditioned on real data. In all cases considered below, a two-level distribution is used, corresponding to a bi-gram language model, but the methodology generalises to more complex models. Where a single value for α is given, the constraint $\alpha_1 = \alpha_2$ is used.

2.7.1 Comparison of the two approximations

The difference between the true distribution given by the hierarchical Dirichlet distribution, and either of the approximations described in Section 2.6 can be measured using the KL divergence. As mentioned above, it is not tractable to exactly compute the probability of data under the hierarchical Dirichlet distribution, so this quantity cannot be used directly. However, given two approximations, Q_1 and Q_2, the difference between their respective KL divergences can be written as

$$\begin{aligned} \mathrm{D_{kl}}\left(P, Q_1\right) - \mathrm{D_{kl}}\left(P, Q_2\right) &= \sum_x P\left(x\right) \log \frac{P\left(x\right)}{Q_1\left(x\right)} - \sum_x P\left(x\right) \log \frac{P\left(x\right)}{Q_2\left(x\right)} && (2.61) \\ &= \sum_x P\left(x\right) \log \frac{Q_2\left(x\right)}{Q_1\left(x\right)} && (2.62) \end{aligned}$$

By sampling values $\{x_i\}_{i=1}^{N}$ from P, this can be approximated by

$$\mathrm{D_{kl}}\left(P, Q_1\right) - \mathrm{D_{kl}}\left(P, Q_2\right) \approx \frac{1}{N} \sum_i \log \frac{Q_2\left(x_i\right)}{Q_1\left(x_i\right)} \qquad (2.63)$$

A positive value for this quantity indicates that Q_2 is the better distribution, whereas a negative value indicates the converse is true.

In the case of the hierarchical Dirichlet distribution, sampling can be done easily by following the generative process, generating probability vectors for each node in the hierarchy and then sampling data from those present in the leaves.

We used this method to evaluate the relative performance of the two approximations. Initially, samples were generated as a sequence — in other words, the context used for each sample was taken to be the value of the previous sample, which is how the model is assumed to be used in the language modelling task. The results of this experiment using 10,000 Monte Carlo samples, each consisting of 1,000 data points, is shown in Figure 2.12.

The results indicate that the minimal oracle approximation does indeed perform better for small values of α, as was suggested in Section 2.6. Furthermore, the maximum value of α for which this is true increases with the alphabet size. This variation is indicated in Figure 2.13 (blue curve).

Interestingly, it appears that the performance of the approximation is dependent on the distribution of contexts. Figure 2.14 shows the performance as a function of α when contexts are chosen at random from a uniform distribution rather than using the previously drawn symbol. The key difference between these two approaches is that the former will tend to produce a non-uniform distribution over contexts, whereas a uniform distribution is assumed

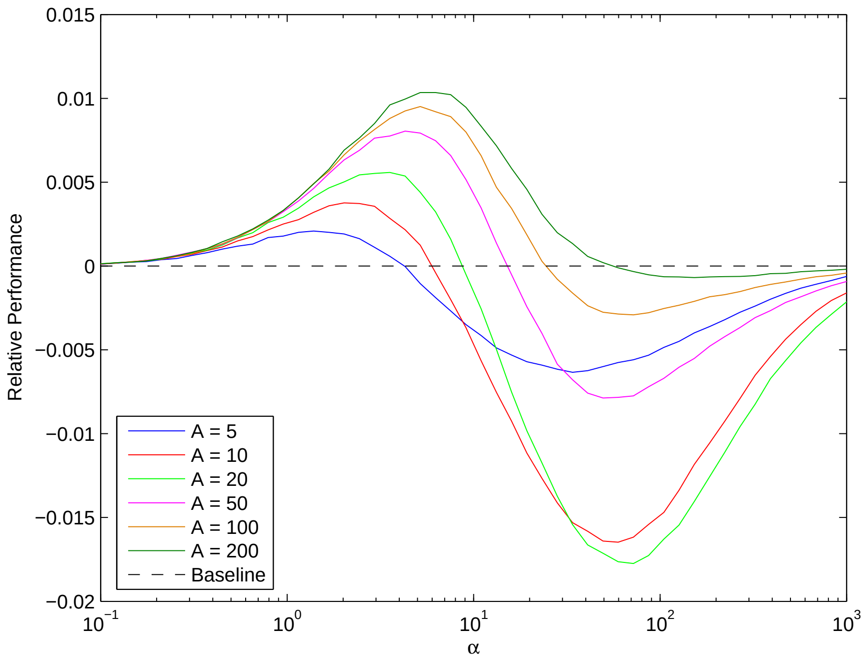


Figure 2.12: Relative performance of the two approximations as a function of alphabet size, A, and the Dirichlet parameter, α, under sequential generation. Results shown are Monte Carlo approximations using 10000 samples of length 1000. Positive values indicate better performance for the minimal oracle approximation.

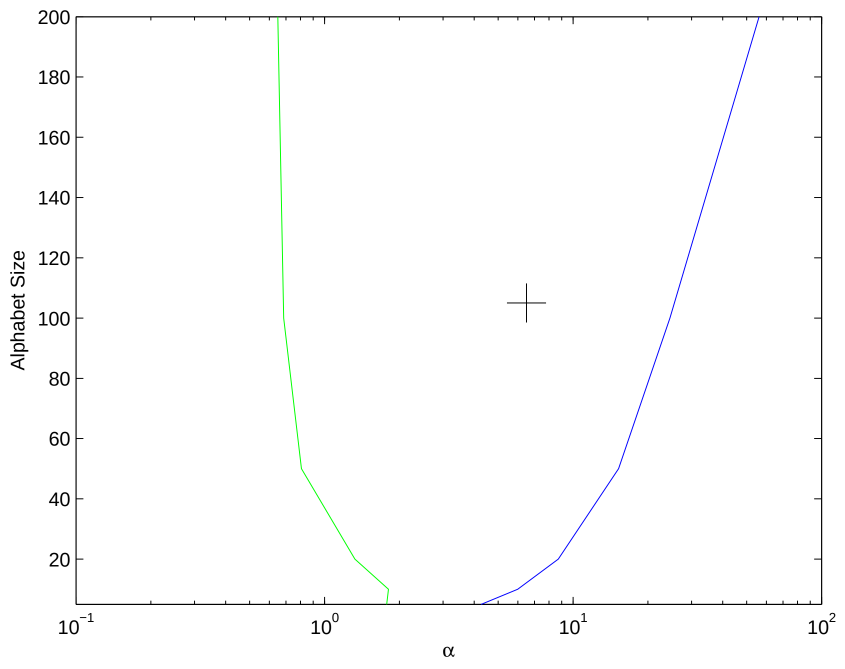


Figure 2.13: Boundary values of α, below which the minimal oracle approximation gives better performance. Curves are for sequential generation (blue) and random contexts (green). The black cross indicates the optimal value of $\alpha = 6.5$ found in Section 2.4 for bi-gram generalised PPM-A, which was for an alphabet size of 105.

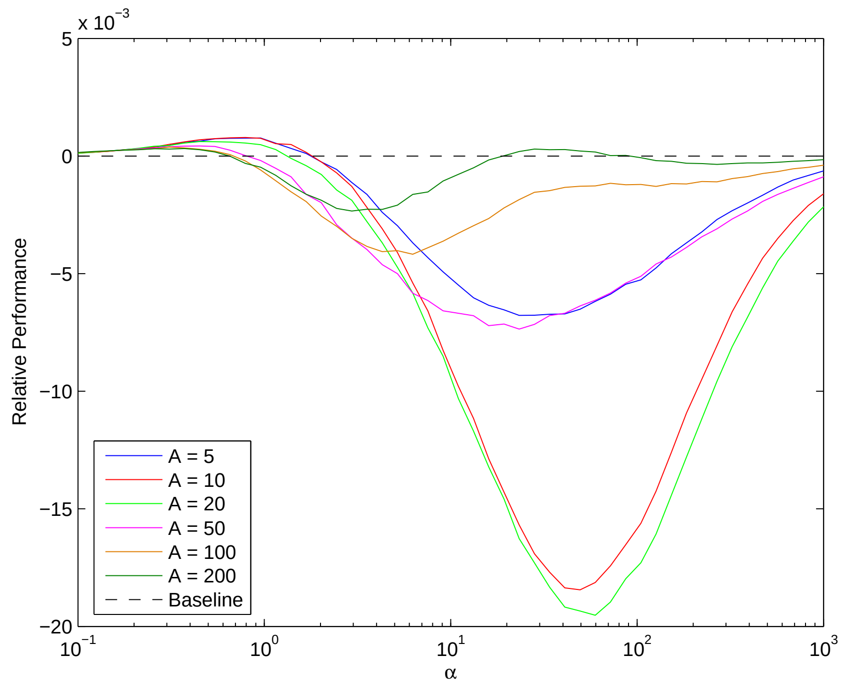


Figure 2.14: Relative performance of the two approximations as a function of alphabet size, A, and the Dirichlet parameter, α, when contexts are drawn at random from a uniform distribution. Results shown are Monte Carlo approximations using 10,000 samples of length 1,000. Positive values indicate better performance for the minimal oracle approximation.

in the latter. In this case, the maximum value of α *decreases* as the alphabet size is increased, although in the case of $A = 200$ a second range of α values where the minimal oracle approximation is superior is observed. Boundary values for a range of values of A are also shown in Figure 2.13 (green curve).

We also considered varying α_1 and α_2 independently, producing the results shown in Figure 2.15 as a contour plot. The figure shows the relative performance of the two approximations as a function of these variables. The zero contour is highlighted, as this represents the boundary between the regions in which each approximation gives the best performance. The minimal oracle assumption gives the better approximation to the left of the plot, corresponding to small values of α_2. The trend with respect to α_2 can be understood once more in terms of the probability at each time step with which an oracle enquiry is made at each time step. The trend with respect to α_1 requires a little more thought, but a possible explanation is related to the fact that this parameter represents the probability that, conditioned on an oracle enquiry being made, a new sample is generated from the top level prior. If this does occur, then the probability of drawing a token which has not previously been seen in the current context is increased. Conditioned on a particular observed token, the probability of an oracle enquiry being made is therefore reduced as α_1 is increased.

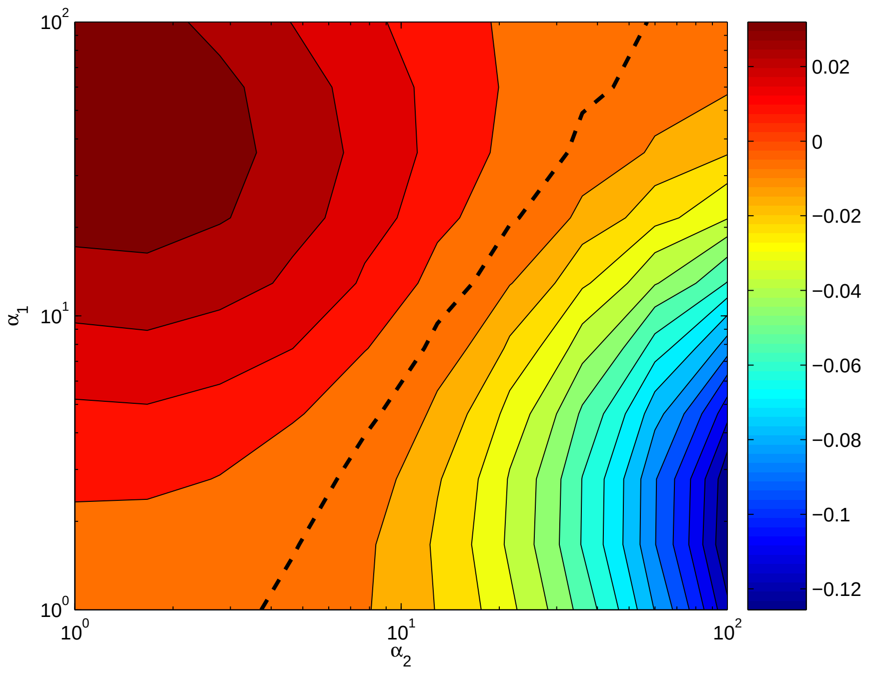


Figure 2.15: Contour plot showing the relative performance of the two approximations as α_1 and α_2 are varied independently. The plots are shown for 10,000 Monte Carlo samples of length 1,000, and an alphabet size of 100. The zero contour is highlighted as a dashed line. To the left of this contour the minimal oracle assumption more closely approximates the true distribution.

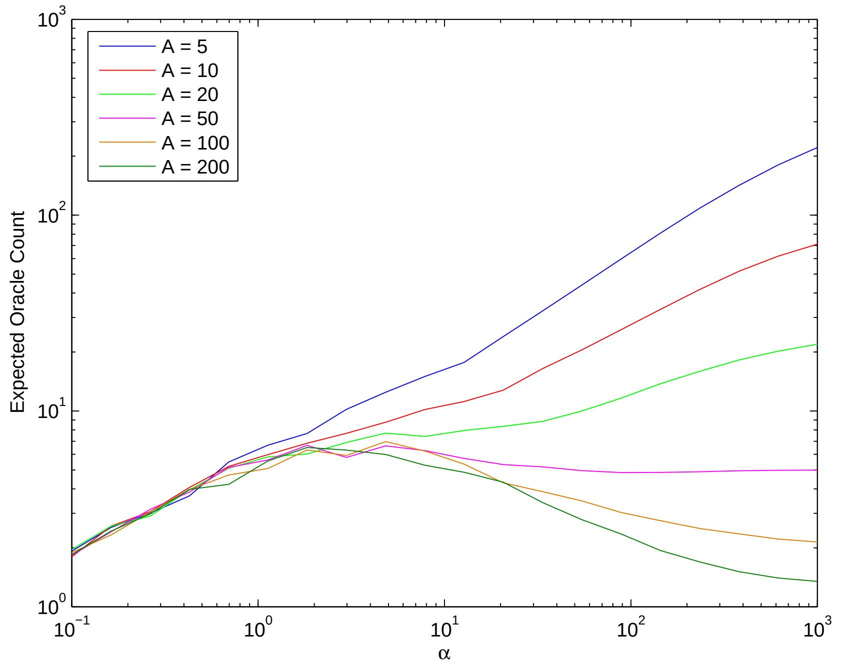


Figure 2.16: Mean number of oracle enquires per group as a function of the alphabet size and α. Results are mean values over ten repetitions, for data sets of length 10,000.

2.7.2 Investigating the prior distribution

As well as investigating the performance of the proposed approximations to the prior distribution, it is worthwhile to characterise the exact distribution, which can also be done by Monte Carlo sampling. By drawing samples from this distribution, we estimated the marginal prior distribution over oracle paths. Let X_{jk} be the set of all locations in the text where token t_j occurs in context $\mathcal{C}_k$. Let n_{jk} be the number of oracle enquiries made within X_{jk}. Recall that the minimal oracle assumption corresponds to the assumption that $n_{jk} = 1$ for all j and k, whereas the maximal oracle assumption corresponds to $n_{jk} = |\mathsf{X}_{jk}|$. Define $\bar{n}$ to be the expected value of n_{jk} weighted according to the size of the corresponding set, in other words,

$$\bar{n} = \frac{\sum_{jk} |\mathsf{X}_{jk}|\, n_{jk}}{\sum_{jk} |\mathsf{X}_{jk}|} \tag{2.64}$$

Intuitively, this corresponds to the expected value of $\bar{n}$ when locations in the text are drawn from a uniform distribution. Figure 2.16 shows the value of $\bar{n}$ for various values of α.

For small token sets, the graph shows a clear trend for $\bar{n}$ to increase with α. As mentioned previously, this is as expected — α can be interpreted as the number of data points which must be seen in a group before the local statistics are trusted. Before this point there is a high probability of oracle enquiries taking place. However, for larger alphabet sizes the expected oracle count reaches a maximum, starting to decline as large values of α are used.

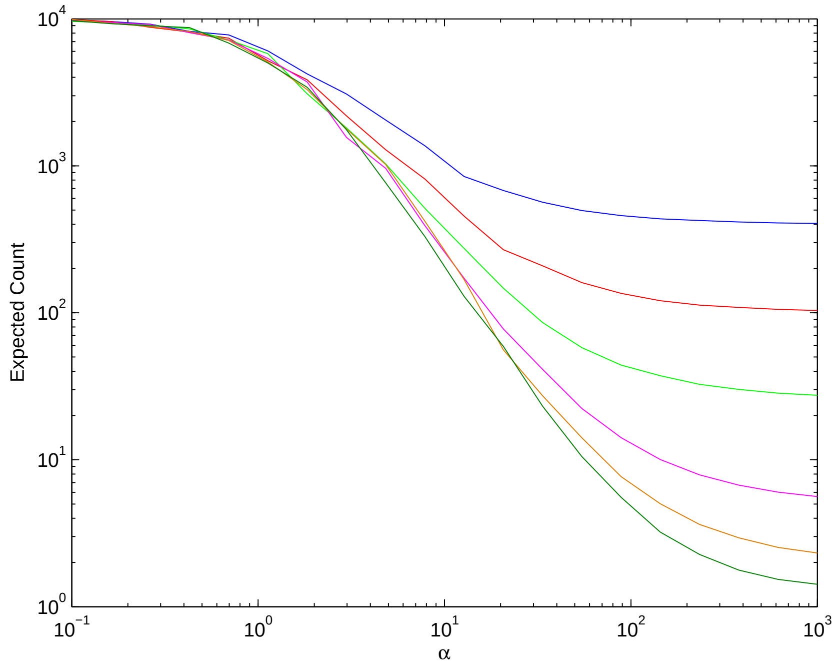


Figure 2.17: Mean number of entries per group as a function of the alphabet size and α. Results are mean values over ten repetitions, for data sets of length 10,000.

This can be understood by looking at the expected value of $|\mathsf{X}_{jk}|$ when averaged in the same way, shown in Figure 2.17. For large α this is reduced, with the reduction being greatest for large alphabet sizes. There are therefore fewer opportunities to ask the oracle in this extreme. Figure 2.18 shows the distribution of n_{jk} as a histogram for an alphabet size of 105 and four values of α. The trend is in keeping with that shown in Figure 2.16.

The results presented in Figures 2.16 and 2.17 do not allow direct comparison with the number of oracle enquiries for each of the proposed approximations. In order to examine the relationship between the prior distribution and these approximations it is useful to consider the evolution of the number of oracle enquiries as samples are generated. To obtain this information, we calculated the number of oracle enquiries after each data point had been generated for a single sample from the prior, with $\alpha = 6.5$ and an alphabet size of 105, corresponding to the values found for optimisation on real text. We also calculated the number which would have occurred under each of the approximations. The results of these calculations are shown in Figure 2.19. The figure clearly shows that the evolution for the prior distribution follows a path which is closer to the minimal oracle approximation than the maximal oracle approximation, giving further evidence for the strength of update exclusion.

Finally, as with the comparison between approximations, we considered the case when α_1 and α_2 are not constrained to take the same value. This is shown in Figure 2.20, indicating that the highest number of oracle enquiries is found for large α_2 and small α_1. This can

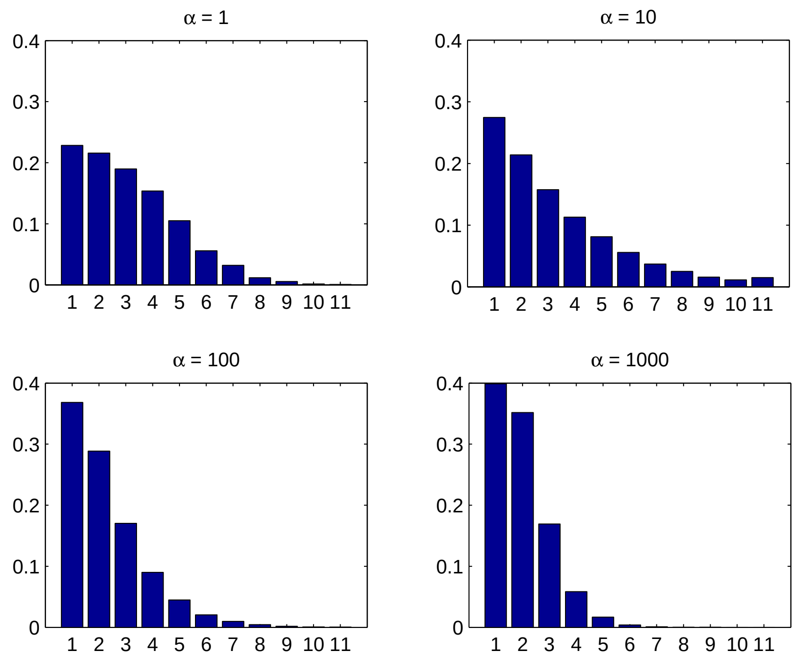


Figure 2.18: Bar charts showing the distribution of group counts for $\alpha = 1$, 10, 100 and 1000. Data is shown for an alphabet size of 105 and a data length of 10,000.

be understood in terms of there being a large probability of making an oracle enquiry, but a small probability of it returning a new value, so it will instead increase the count for an existing group. The cross-hair again indicates $\alpha_1 = \alpha_2 = 6.5$, the optimal value found for generalised PPM-A in Section 2.4. The expected oracle enquiry count in this case is around 7, whereas the expected group count is nearer to 1,000 (see Figure 2.17), again indicating that the minimal oracle assumption is likely to be relatively good.

2.7.3 Posterior distribution

While exploring the prior gives a valuable insight into the behaviour of the language model, the distribution is necessarily a simplification of the true processes which generated the text, and the behaviour conditioned on data may be quite different. To show this, Figure 2.21 uses Hinton diagrams to represent two samples from the hierarchical Dirichlet distribution with $\alpha_1 = \alpha_2 = 6.5$ as well as the true bi-gram statistics from an example of English text. The latter is markedly different, most strikingly in terms of the division of the alphabet into blocks, which correspond to lower case letters, upper case letters, numerals and so on, and which have very different statistics.

In order to explore the posterior distribution, a slightly more sophisticated Monte Carlo approach is required. Let an index be associated with each enquiry, and let $\{z_i\}$ be unobserved variables associated with each data point. In the case of sequential generation of data, this

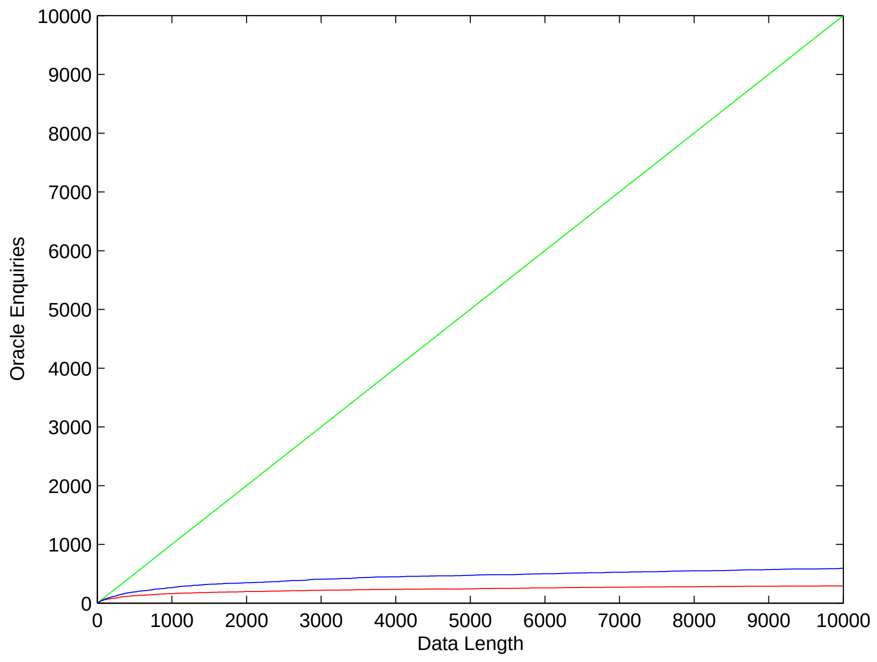


Figure 2.19: Number of oracle enquiries as a function of the number of data points generated. The blue curve shows the evolution of a typical sample from the prior distribution, whereas the red and green curves show the minimal and maximal oracle assumptions respectively.

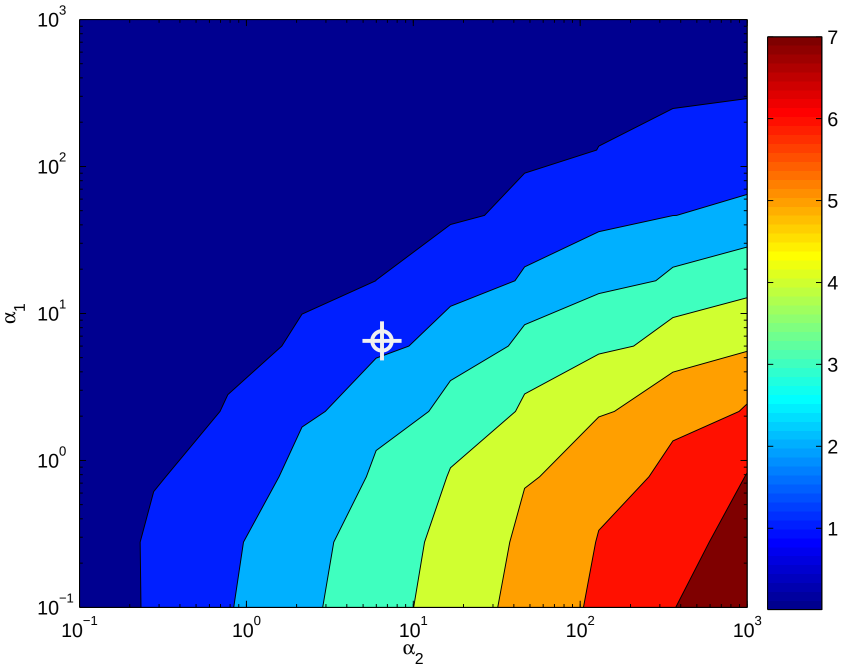
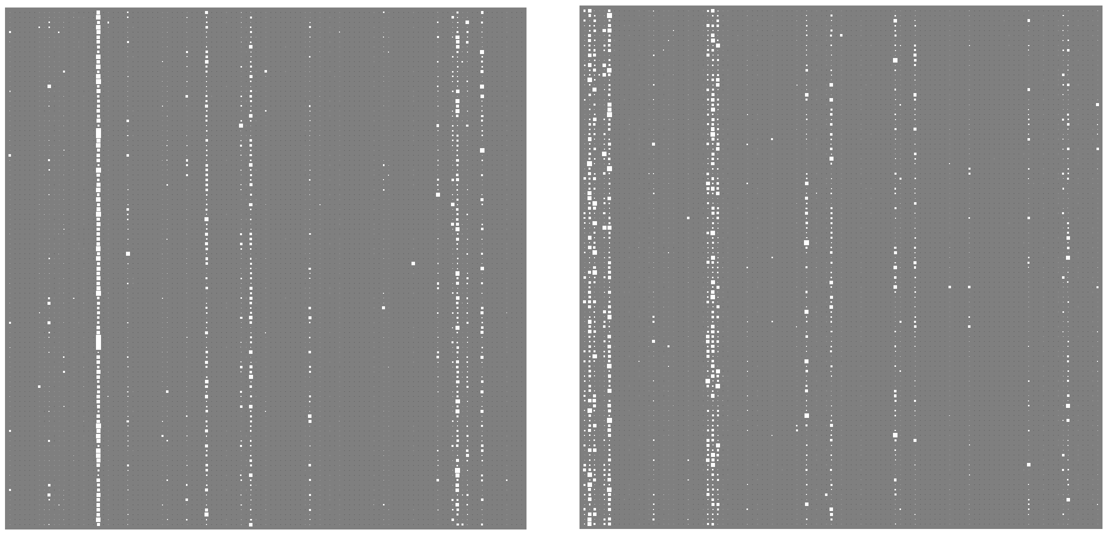
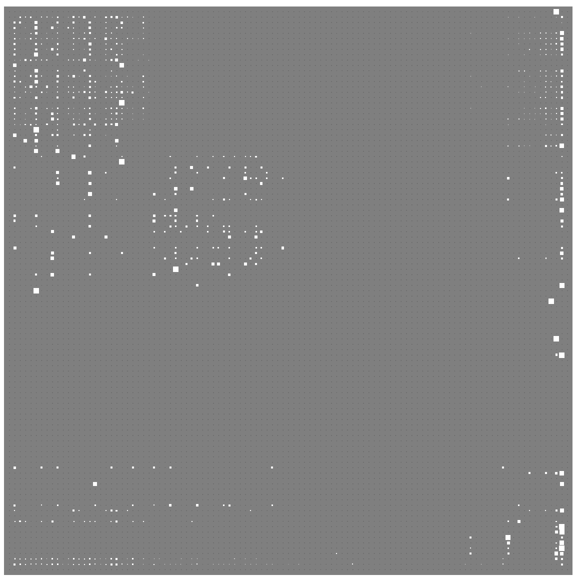


Figure 2.20: Mean number of groups (shown as logarithmic values) as a function of both α parameters for fixed alphabet size of 105. $\alpha_1 = \alpha_2 = 6.5$, the experimentally found values for generalised PPM-A, is indicated by a cross-hair.



(a)



(b)

Figure 2.21: (a): Hinton diagrams for samples from the hierarchical Dirichlet distribution with an alphabet size of 105 and $\alpha = 6.5$ (The values found by experimental optimisation of the generalised PPM-A parameters on real text). (b): Maximum likelihood bi-gram conditional probabilities for 10kB of real text. Hinton diagrams represent each element of a matrix by a square, the size of which is proportional to the magnitude of the element. White squares indicate positive values, black squares (of which there are none here) indicate negative values. The matrices represent the conditional distributions - in other word, the ij^{th} element is the probability of token t_j occurring conditioned on the context being $\mathcal{C}_i$. Notice the structure present in (b) as a result of the division of the alphabet into upper and lower case letters, punctuation and so on which is missing in (a).

index is used as follows: If an oracle enquiry is made when the data point is generated, then a new, unique value is assigned to the z variable associated with new data point. If the oracle is not asked, then a previously observed sample is selected as described in Figure 2.11. However, when making the selection a distinction is made between previously observed samples when they have a different z value, even if they resulted in the same token being produced. This is possible due to the fact that probability of selecting each previously observed value is proportional to the number of times it has been observed. The value of the z variable associated with the newly generated sample is then set to be equal to that associated with the previous sample selected to generate it. Each z value may be viewed as identifying a specific oracle enquiry to which a given data point belongs, and the number of active value of z therefore corresponds to the number of enquiries which were made. Given a set of observed tokens, samples can be obtained from the posterior distribution over $\boldsymbol{z}$ using Gibbs sampling [56]. The Gibbs sampling algorithm proceeds as shown in Algorithm 1, where the notation $\boldsymbol{z}^{\setminus i}$ is used to represent values excluding data point i.

Algorithm 1 Gibbs sampling in the hierarchical Dirichlet model

```
1: Set $n_{max}$ to be the desired number of samples.
2: Pick an initial value, $\boldsymbol{z}$.
3: for $n = 1$ to $n_{max}$ do
4:   for $i = 1$ to $M$ do
5:     Pick $z_i \sim P\left(z_i \mid \boldsymbol{z}^{\setminus i}, \boldsymbol{t}\right)$.
6:   end for
7:   Return $\boldsymbol{z}$ as a new sample.
8: end for
```

Samples from the hierarchical Dirichlet distribution are exchangeable. In other words, the probability is invariant under permutation of the samples. When sampling from the conditional distribution, it is therefore possible to calculate the conditional distribution as if the particular data point being updated is the last one to be generated. The conditional distribution is therefore

$$\Pr\left(z_i \mid \boldsymbol{z}^{\setminus i}, \boldsymbol{t}\right) \propto \begin{cases} n_t^{\setminus i}\left(z_i\right) & \text{If } z_i \text{ is an existing enquiry} \\ \alpha_2 \cdot \dfrac{n_z^{\setminus i}\left(t_i\right) + \frac{\alpha_1}{|T|}}{N_z^{\setminus i} + \alpha_1} & \text{If } z_i \text{ is a new enquiry} \end{cases} \tag{2.65}$$

where $n_t\left(z_i\right)$ is the number of data points associated with oracle enquiry z_i, $n_z\left(t_i\right)$ is the total number of distinct values of z associated with data points equivalent to t_i (i.e. the same token in the same context), and N_z is the total number of oracle enquiries in the data set.

2.7.4 Sampling results

We performed three experiments using samples from the posterior distribution. Firstly, we considered the mean number of oracle enquiries per token/context combination for $\alpha = 6.5$

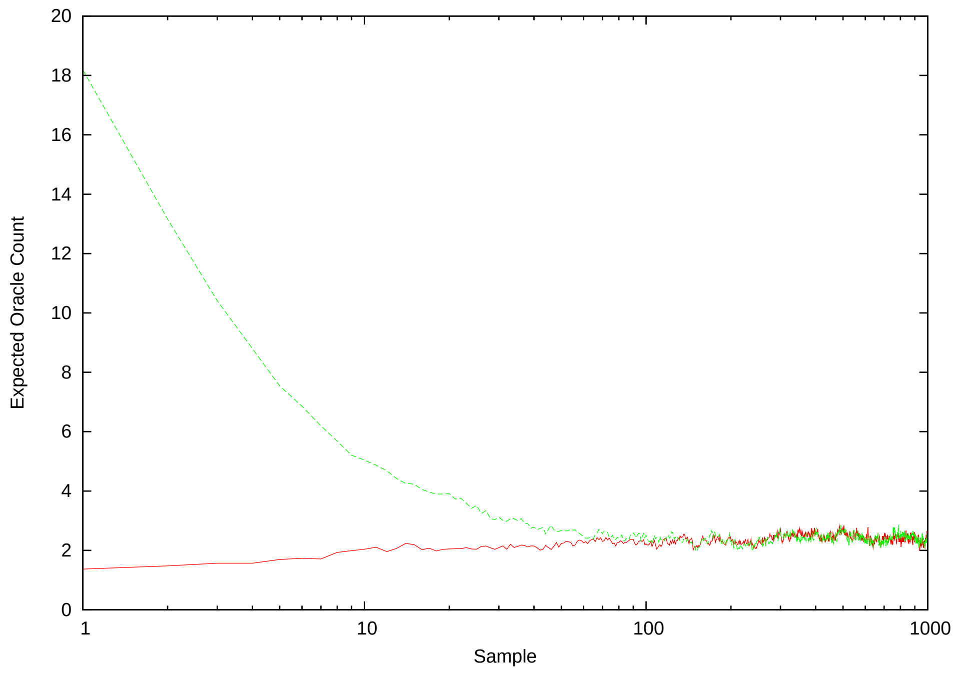


Figure 2.22: Samples from the posterior distribution, starting from both the minimal and maximal oracle assumptions. The average value over 800 iterations, allowing 200 iterations for burn in is 2.41 ± 0.01 in both cases.

and $A = 105$. The value of this quantity after each sample is shown in Figure 2.22. The main purpose of this experiment was to verify that the Monte Carlo algorithm was working correctly. We used two starting configurations, one starting with one oracle enquiry for each data point (corresponding to the maximal oracle assumption), and one assigning a single oracle enquiry for each symbol in each context (corresponding to the minimal oracle assumption). Figure 2.22 shows that the two starting points converge after a little over 100 samples, indicating that the algorithm does indeed work correctly, and giving a figure for the burn-in time required in order to obtain samples which are independent of the initial conditions. The expected value of around 2.4 oracle enquiries per token/context pair is actually less than was expected from the prior, indicating that the minimal oracle approximation might be better than the prior experiments alone suggest.

As with the prior distribution, we also investigated the evolution of the number of oracle enquiries after each data point. The results of this experiment are shown in 2.23, which shows ten samples from the posterior as well as the minimal and maximal oracle assumptions. Again, the figure indicates that the minimal oracle assumption gives the better approximation to the true path.

More usefully, having obtained samples from the posterior distribution over oracle paths, it is possible to estimate the information rate under the hierarchical Dirichlet model. We calculated the mean information rate over the posterior samples, which is shown in Figure 2.24 as a function of α, together with generalised PPM-A both with and without update

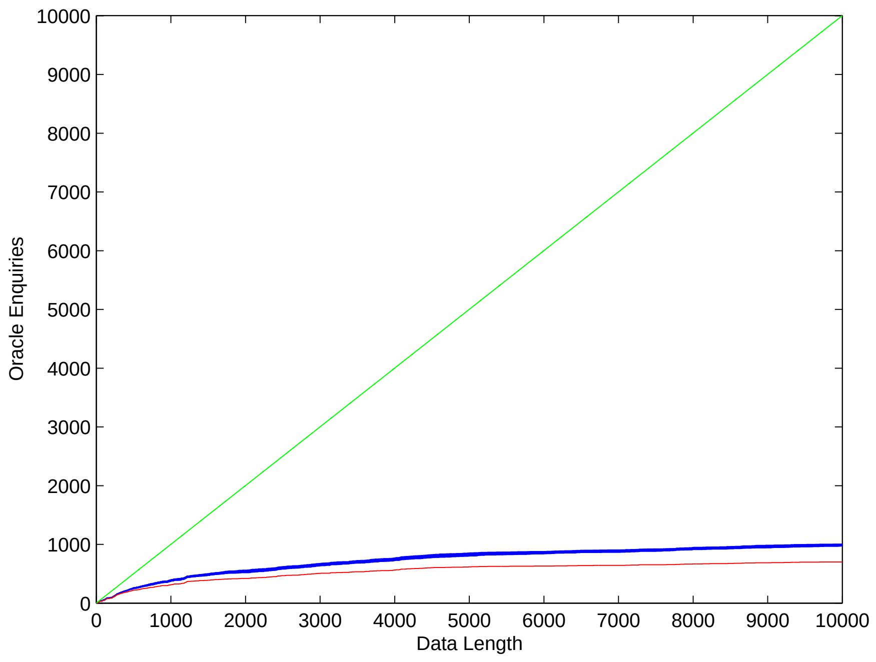


Figure 2.23: Number of oracle enquiries as a function of the number of data points generated. The blue curve shows the evolution of ten samples from the posterior distribution, whereas the red and green curves show the minimal and maximal oracle assumptions respectively.

exclusion. The results show that there is essentially no difference between the achievable information rates with the hierarchical model and with generalised PPM-A with update exclusion (although the values of α for which the optimum is obtained do vary slightly). This result suggests that the minimal oracle assumption provides an excellent approximation to the hierarchical model in this case. For larger values of α the maximal oracle assumption more closely approximates the performance of the hierarchical model, which is consistent with results presented earlier in this chapter. In general, the hierarchical model provides a lower envelope for the information rate of the two approximations.

2.8 A note on the two parameter model

The Dirichlet distribution, or more precisely, the Dirichlet process which is the infinite equivalent, is a special case of the Pitman-Yor process [65]. The predictive distribution, which in this case is over a continuous probability space, is given by

$$P\left(x_{M+1} \mid x_1, \ldots, x_M\right) = \sum_{i=1}^{N} \frac{m_i - \beta}{M + \alpha} \delta_{x_i} + \frac{N\beta + \alpha}{M + \alpha} P_0\left(x\right) \tag{2.66}$$

where x_i are previous samples, each of which has been seen m_i times, and P_0 is a base distribution. α and β are parameters of the distribution with $\alpha > 0$ and $0 \leq \beta < 1$. In the special case of $\beta = 0$, the Dirichlet process is recovered, whereas the case $\alpha = 0$ gives a prediction rule similar to that used in Kneser–Ney smoothing (see Section 2.3.7). The similarity between Kneser–Ney smoothing and the Pitman–Yor process has also been noted by Goldwater *et. al.* [28] and by Teh [91].

2.9 Conclusions

This chapter considered the relationship between generalised PPM-A and the hierarchical Dirichlet language model. It was shown that it is possible to view the former as an approximation to the latter, with the choice between update exclusion and using full counts being one of the decisions affecting the nature of the approximation. Monte Carlo simulations were used to analyse the applicability of the two approximations, revealing that in typical conditions that corresponding to update exclusion is closer to the hierarchical Dirichlet model. Combined with previous empirical results indicating that update exclusion typically leads to an improvement in performance, this result reflects favourably on the hierarchical model. Further experiments were performed to sample from the posterior distribution under the hierarchical Dirichlet model, showing that performance appears to very closely comparable to generalised PPM-A using update exclusion.

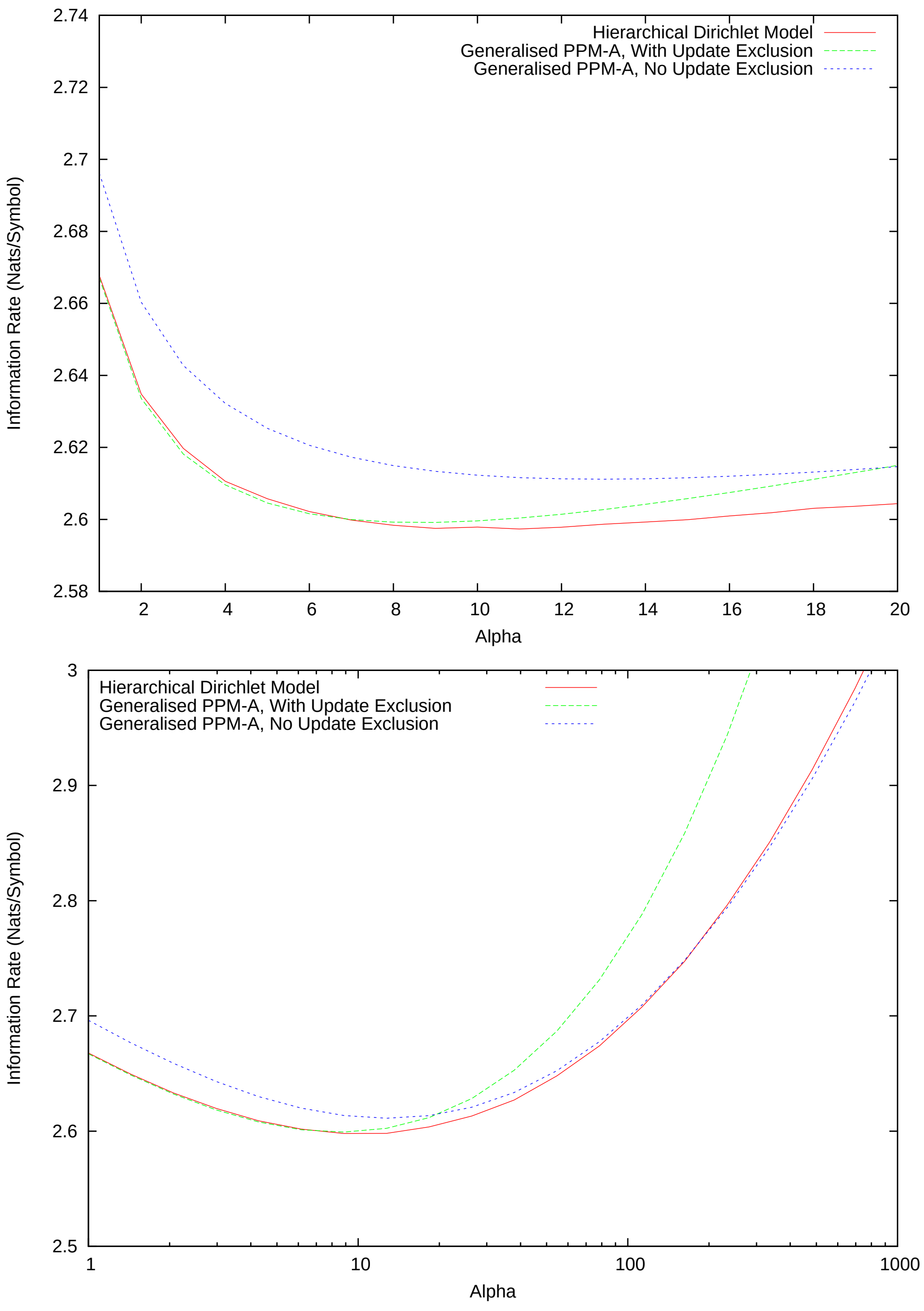


Figure 2.24: Comparison of the full hierarchical model to generalised PPM-A, using Monte Carlo simulation. Top: Detail of the region close to optimal performance. Bottom: Variation over a wider range of α, shown on a logarithmic scale.