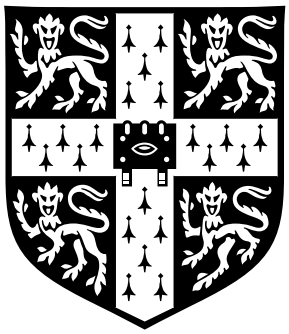
Probabilistic Document Modelling

Philip J. Cowans
Churchill College
Cambridge

A dissertation submitted in candidature for the degree of Doctor of Philosophy,
University of Cambridge

Inference Group
Cavendish Laboratory
University of Cambridge



January 2006

Declaration

I hereby declare that my dissertation entitled "Probabilistic Document Modelling" is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other University.

I further state that no part of my dissertation has already been or is being concurrently submitted for any such degree or diploma or other qualification.

Except where explicit reference is made to the work of others, this dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration. This dissertation does not exceed sixty thousand words in length.

Date: ..... Signed: .....

Philip J. Cowans
Churchill College
Cambridge
January, 2006

Abstract

In this thesis the development and application of probabilistic models of documents is considered. The initial focus is on language models which provide a way of modelling plain text documents. In particular the hierarchical Dirichlet language model, which is derived from simple Bayesian theory, is investigated and is shown to be well approximated by an existing method known as generalised PPM-A. Using this equivalence, generalised PPM-A is extended to produce a language model which while working on the level of individual letter-like symbols is able to make use of the division of the text stream into words. It is shown that the new model can be used in conjunction with a word list to improve performance when very little information from which to learn the statistics of the language is available.

The hierarchical Dirichlet model is then applied to the task of information retrieval, producing a new retrieval method which naturally includes document frequency information. This information has traditionally been used in retrieval systems, but previously had either been missing or introduced heuristically in language model based approaches to the problem. The hierarchical approach is also extended to the task of retrieval at the passage level where it is shown to give promising results.

Finally, the scope of the investigation is broadened to include documents which contain diagrams as well as plain text. A method is developed to group fragments of digitised ink strokes into perceptually relevant components of a diagram, while at the same time labelling the components with an object class. The approach, which is based on the conditional random field, is shown to work well both in terms of grouping and improving labelling performance when compared to other methods.

Acknowledgements

I would like to thank Ryan Adams, Tom Minka, Iain Murray, Edward Ratzer, Oliver Stegle, David Stern, Martin Szummer, Keith Vertanen, Hanna Wallach, David Ward, Seb Wills and John Winn for ideas, suggestions, discussions and feedback without which the work presented in this thesis would not have been possible.

During my studies I was lucky enough to be able to spend three months working in the Machine Learning and Perception group at Microsoft Research UK, an opportunity for which I am grateful and which produced many of the ideas presented in Chapter 6.

Above all, I would like to thank my supervisor, Professor David MacKay for an unending supply of inspiration, discussion and feedback, and for providing invaluable advice on the draft of this thesis. I would also like to thank my parents, Sue and Trevor Cowans, and my brother, Nick, for their support and encouragement during the last four years.

This work was supported by a grant from Microsoft Corporation.

Contents

Chapter 1

Introduction

We live in a world where information increasingly exists in an electronic form, stored on computer systems which through the internet are accessible across the planet. Recent estimates suggest that there are over 11.5 billion pages in the indexable portion of the world wide web [30], and a number of projects are digitising large quantities of material which currently only exists in printed form [102] [101].

The availability of such large quantities of information presents opportunities and challenges. The work presented in this thesis covers three areas:

- **Acquisition of information:** This area includes topics such as text entry and speech recognition which are used to directly create new works, or areas such as optical character recognition which convert existing information into an electronic form.
- **Processing of information:** For example, interpretation, summarisation and searching for electronic resources.
- **Presentation of information:** When a user has requested information, how should it be presented to them?

The unifying theme will be the use of *probability theory* to perform these tasks.

1.1 Why probabilistic models?

The subjective interpretation of probability views probabilities as statements of *degrees of belief* in hypotheses [24]. Probability theory can be seen as an extension of classical Boolean logic [34] — a set of axioms for making consistent statements about our beliefs.

Probability theory allows simple models to be constructed which are able to capture many of the interesting properties of complicated systems. Most documents are the product of human thought, complex social interactions and the workings of large physical systems. Attempting to understand such documents by building models which mimic these processes is well beyond our current abilities, so the power afforded by simple probabilistic models is invaluable.

Perhaps an even more important advantage of probabilistic modelling is the possibility of *learning*. In other words the details of a model need not be fully specified when it is created and behaviour can be learned by example. In the probabilistic framework, learning is achieved by jointly modelling all observations. Statistical correlations are assumed to exist between what has been observed in the past and what will be observed in the future. This framework gives rise to the idea of *prior* and *posterior* distributions, the latter of which encapsulates current beliefs in the underlying operation of the world, and can be updated using Bayes' theorem.

1.2 Fundamentals of probability theory

Let x be a random variable. The probability distribution over x is written $\Pr(x)$, which is non-negative for all x and is normalised so that

$$\int \mathrm{d}x \Pr(x) = 1 \tag{1.1}$$

Two random variables, x and y can be expressed in terms of a *joint probability distribution*, $\Pr(x, y)$, which can encapsulate any correlations between their values. Given such a distribution, the marginal distribution over x is given by

$$\Pr(x) = \int \mathrm{d}y \Pr(x, y) \tag{1.2}$$

with an equivalent definition for the marginal distribution over y. The conditional probability distribution over x when y is observed is given by

$$\Pr(x \mid y) = \frac{\Pr(x, y)}{\Pr(y)} \tag{1.3}$$

If $\Pr(x \mid y) = \Pr(x)$ then x and y are said to be independent. The joint probability can be decomposed in two ways,

$$\Pr(x, y) = \Pr(x \mid y) \cdot \Pr(y) \tag{1.4}$$

and

$$\Pr(x, y) = \Pr(y \mid x) \cdot \Pr(x) \tag{1.5}$$

Equating these gives

$$\Pr(x \mid y) \cdot \Pr(y) = \Pr(y \mid x) \cdot \Pr(x) \tag{1.6}$$

$$\Pr(x \mid y) = \frac{\Pr(y \mid x) \cdot \Pr(x)}{\Pr(y)} \tag{1.7}$$

Which is Bayes' famous rule. One particularly useful application is the case where multiple observations, $\{x_i\}_{i=1}^N$ are jointly distributed with some parameters, y. In this case,

$$\Pr(x_N \mid x_1, \ldots, x_{N-1}) = \int \mathrm{d}y \Pr(x_N, y \mid x_1, \ldots, x_{N-1}) \quad (1.8)$$

$$= \int \mathrm{d}y \Pr(x_N \mid y, x_1, \ldots, x_{N-1}) \cdot \Pr(y \mid x_1, \ldots, x_{N-1}) \quad (1.9)$$

If $\{x_i\}_{i=1}^N$ are independent given y, then

$$\Pr(x_N \mid y, x_1, \ldots, x_{N-1}) = \Pr(x_N \mid y) \quad (1.10)$$

which gives

$$\Pr(x_N \mid x_1, \ldots, x_{N-1}) = \int \mathrm{d}y \Pr(x_N \mid y) \cdot \Pr(y \mid x_1, \ldots, x_{N-1}) \quad (1.11)$$

De Finetti's theorem states that this conditional independence assumption holds if and only if $\{x\}$ are exchangeable, that is, if their probability is invariant under permutation. Equation (1.11) is the basis of Bayesian machine learning. The term $\Pr(y \mid x_1, \ldots, x_{N-1})$ is the *posterior* distribution over y given the data seen so far, and can be calculated using Bayes' theorem,

$$\Pr(y \mid x_1, \ldots, x_{N-1}) = \frac{\Pr(x_1, \ldots, x_{N-1} \mid y) \cdot \Pr(y)}{\Pr(x_1, \ldots, x_{N-1})} \quad (1.12)$$

where $\Pr(x_1, \ldots, x_{N-1} \mid y)$ is the *likelihood* and $\Pr(y)$ is the *prior*. Intuitively speaking, the prior represents our belief about y before any data is seen, and the posterior represents the belief after seeing the data. As more data is observed, the posterior will typically become tighter as more is learnt.

1.3 Applications of document modelling

The subject of this thesis is the creation and application of models of documents expressed in terms of probability distributions. There are many areas in which this approach has been shown to be fruitful, some of which are briefly introduced in the following sections.

1.3.1 Language modelling

The majority of the work in this thesis concerns plain text documents, with the aim being to produce language models consisting of probability distributions over strings of tokens. These distributions represent our belief in the likely content of newly observed documents. Chapters 2 and 3 of this thesis consider language modelling from a Bayesian viewpoint.

Application	Observation	Source
Speech recognition	Acoustic signal	Spoken text
Handwriting recognition	Digitised pen strokes or bitmap image	Written text
Optical character recognition	Bitmap image	Written text
Automatic translation	Text in source language	Text in target language
Spelling correction	Mis-spelt word	Correct spelling
Ambiguous text entry	Keystroke sequence	Entered text

Table 1.1: Natural language applications of channel modelling.

Channel models

Consider a noisy communications channel, which transmits some stochastically generated source signal $\boldsymbol{y}$. An observation, $\boldsymbol{x}$, is made of the resulting output of the channel, with the goal being to reconstruct $\boldsymbol{y}$. In order to perform such a reconstruction it is necessary both to have a model of the channel, representing beliefs in the transformations which may have occurred, and a model of the source.

Formally speaking, the posterior distribution over $\boldsymbol{y}$ can be found using Bayes' rule,

$$\Pr(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\Pr(\boldsymbol{x} \mid \boldsymbol{y}) \Pr(\boldsymbol{y})}{\Pr(\boldsymbol{x})} \tag{1.13}$$

$\Pr(\boldsymbol{x} \mid \boldsymbol{y})$ is the model of the channel, which is application dependent. $\Pr(\boldsymbol{y})$ is the prior distribution, representing our belief in the input in the absence of any observations.

This framework may be applied to a number of problems involving natural languages, for example speech recognition [36] [70], handwriting recognition [69], optical character recognition, automatic translation [15], spelling correction [41] and ambiguous text entry. Examples of the channels which are used are given in Table 1.1. In all of these applications the source is a string in a natural language, so the prior distribution is a language model.

Data compression

Data compression is the task of converting input strings into codewords with the aim of reducing the average length of transmitted information. Arithmetic coding makes use of a probabilistic model of input strings, $\hat{p}(x)$, giving an estimate of the probability that x will be selected [98]. By ordering the possible strings in dictionary order, this model can be used to assign non-overlapping intervals on the real line $[0, 1)$.

To specify codewords, intervals on the line are expressed as strings of binary digits. Treating the strings as binary fractions after the decimal point, .0 and .1 divide the line in half, .00, .01, .10 and .11 divide it into quarters and so on. After n binary digits the line is divided

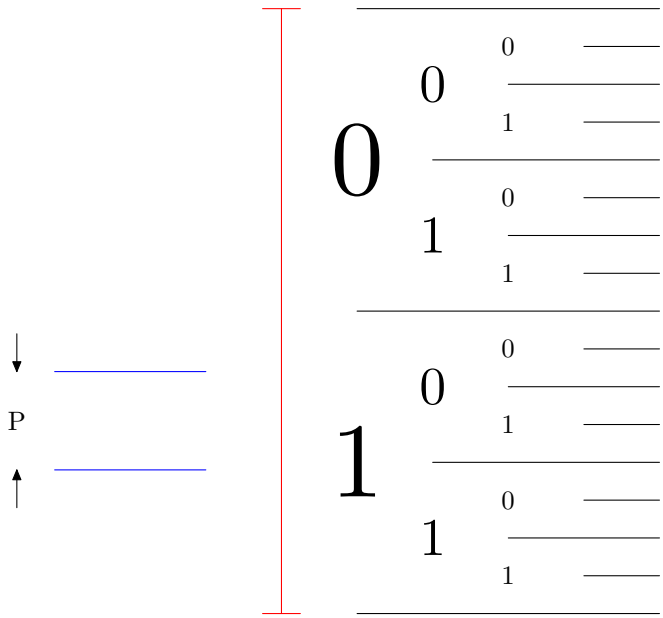


Figure 1.1: An illustration of arithmetic coding. The interval shown in blue on the left hand side corresponds to the string being coded, which has probability P. The right hand side shows the division of the real line (shown in red) into intervals specified by binary codewords. The codeword is the largest binary interval entirely enclosed by the string, in this case 101.

into 2^n intervals of length 2^{-n}. The codeword is the shortest string which is entirely enclosed by the interval defined by the input. Figure 1.1 illustrates the process.

Assuming that the interval corresponding to the string being compressed, x, is exactly aligned with the binary code word intervals,

$$2^{-n} = \hat{p}(x) \tag{1.14}$$

$$n = -\log_2(\hat{p}(x)) \tag{1.15}$$

Taking the expectation with respect to the true distribution over inputs gives

$$\langle n \rangle = -\sum_x p(x) \log_2(\hat{p}(x)) \tag{1.16}$$

where the notation $\langle \cdots \rangle$ is used to represent the expected value of a random variable. If the model exactly matches the source distribution, this expression is equal to the entropy of the source [56], which is the theoretical limit on the compression which can be achieved.

In practice the intervals will not precisely align with the input strings. By considering the worst case, it can be shown that the maximum overhead is 2 bits. As this result holds for arbitrarily long strings, arithmetic compression is able to come very close to ideal performance.

Arithmetic coding essentially reduces the problem of data compression to that of source modelling. If the data to be compressed is a plain text document then the source model is a

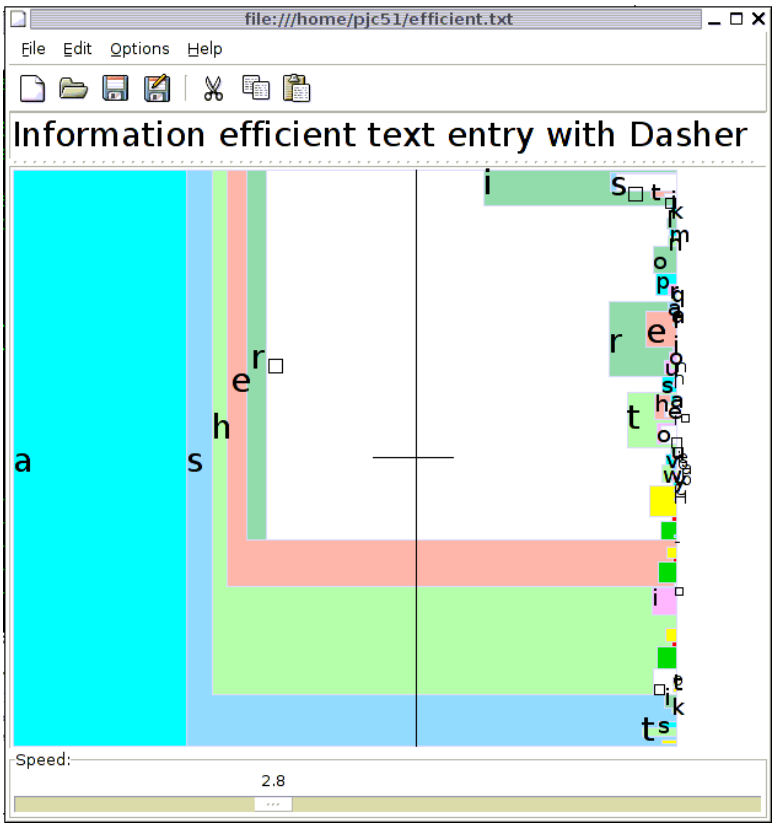


Figure 1.2: The Dasher text entry system. Possible strings are represented as a sequence of nested boxes, with the size of each box being proportional to the conditional probability of each symbol given the parent. The display zooms continuously towards a point on the real line indicated by the user, resulting in the selection of the string corresponding to that point. As a language model is used, more probable strings represent larger intervals on the line, and therefore can be entered more rapidly.

language model of the sort considered in this thesis [6].

Dasher

Dasher is a text entry system originally developed by Ward and MacKay [96] [105] [95]. A screen capture of the Dasher interface is shown in Figure 1.2.

Dasher is based on arithmetic decompression. The Dasher interface evolves dynamically to zoom in on a point on the line, which is equivalent to specifying a codeword. A probabilistic language model is used to model the input strings, which are represented by a series of nested boxes adjoining the line. The user of Dasher can therefore navigate one symbol at a time, with more probable strings being assigned more screen space and therefore being easier to enter.

Dasher is targeted primarily at two areas: firstly, text entry on small devices such as Personal Digital Assistants (PDAs) which are too small to include a usable full size keyboard,

and secondly as a communication aid for people who are prevented from using a normal keyboard by disability. Various adaptations have been used to enable input from eye or head tracking [97], one dimensional input devices such as breath controllers, and button input. Dasher has been adapted for use in a large number of different languages, and facilities exist for using Dasher to control other applications running on the computer. Dasher is also the basis for a novel multi-modal text entry system which combines speech recognition [93]. Dasher is Free software, available under the GNU General Public License. Chapter 5 considers a new use for Dasher — as an efficient method for performing search.

Document classification

The goal of document classification is to group documents which belong to the same class. In many cases the number of classes is fixed and is specified in advance. Let $\{\mathcal{H}_i\}$ be a set of hypotheses concerning the correct classification of a document. By producing a model for the data conditioned on each of these hypotheses, Bayes' rule can be used to compute the posterior probability of class membership,

$$\Pr\left(\mathcal{H}_i \mid \boldsymbol{x}\right) \propto \Pr\left(\boldsymbol{x} \mid \mathcal{H}_i\right) \cdot \Pr\left(\mathcal{H}_i\right) \tag{1.17}$$

Again, we have introduced here a prior, $\Pr\left(\mathcal{H}_i\right)$, on class membership. By comparing the probabilities of membership of each class, decision theory can be used to optimally classify the document.

A topical example of an application of this technique is the task of identifying unsolicited bulk e-mail messages (spam) [80]. By training a model on a large number of examples, reliable classification can be achieved. Further information can be included by adding a vector of features, for example representing details of the message headers. These can simply be treated as part of the document text and modelled in the same way. A typical approach is to use a naïve Bayes classifier, which models documents as 'bags of words', ignoring correlations between terms.

Information retrieval

The task of information retrieval is to search a collection of documents for those which are relevant to a query supplied by the user. A variety of methods have been used for information retrieval, including many which do not make use of probabilistic methods. However, recent research has re-cast the problem in terms of document modelling. Information retrieval has become a significant area for the application of document modelling techniques. Search engines such as Google [107], Yahoo! [114] and MSN Search [110] are now the first port of call for many web users. Information retrieval as an application of language modelling is considered in Chapter 4.

1.3.2 Multimedia documents

Although language modelling is an important aspect of document modelling, it is not the whole story. Many documents contain information beyond that which is contained in the text itself, ranging from the layout of text on the page, through graphical elements to audio and video content.

A particular application beyond language modelling which is considered in this thesis is the analysis of hand-drawn electronic ink diagrams. These diagrams are typically produced using a system which incorporates a pen-based interface, such as a Table PC or a Personal Digital Assistant. The task which will be addressed is the process of identifying perceptually relevant objects in these diagrams. This analysis potentially permits more natural interaction when editing the diagram and allows automatic construction of a more formally laid-out version of a rough sketch. Chapter 6 considers this problem, developing a model based on the conditional random field [48] which is able to simultaneously group ink fragments and label the objects which they represent.

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

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

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

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),

$$R_I = -\frac{1}{M}\sum_{\boldsymbol{t}} P(\boldsymbol{t}) \log_2 \frac{Q(\boldsymbol{t}) \cdot P(\boldsymbol{t})}{P(\boldsymbol{t})} \tag{2.6}$$

$$= \frac{1}{M}\sum_{\boldsymbol{t}} P(\boldsymbol{t}) \log_2 \frac{P(\boldsymbol{t})}{Q(\boldsymbol{t})} - \frac{1}{M}\sum_{\boldsymbol{t}} P(\boldsymbol{t}) \log_2 P(\boldsymbol{t}) \tag{2.7}$$

$$= D_{KL}(P, Q) + H(P). \tag{2.8}$$

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:

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

$$= \frac{m_{jk}}{p_{jk}} + \lambda_k \tag{2.13}$$

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

$$p_{jk} = -\frac{m_{jk}}{\lambda_k} \tag{2.14}$$

$$= \frac{m_{jk}}{\sum_{k'} m_{jk'}} \tag{2.15}$$

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

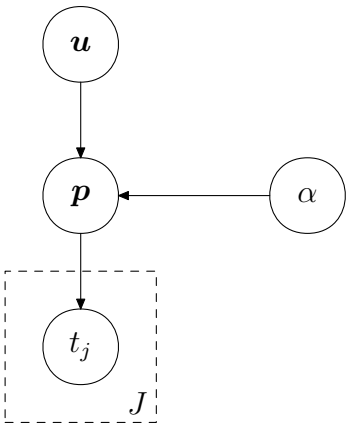


Figure 2.1: A Bayesian network representing a simple language model. Data observed in different contexts are modelled as being independent, with the distribution shown here representing the model for a particular context, $\mathcal{C}$. Each node in the graph represents a variable in the model, with 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 $\alpha =$

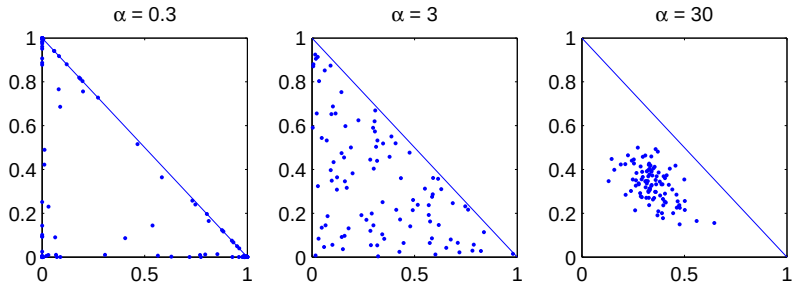


Figure 2.2: Samples from a three dimensional symmetric Dirichlet distribution. From left to right plots are shown for $\alpha = 0.3$, $\alpha = 3$ and $\alpha = 30$. The axes show the first two components of the sampled probability vectors, with the third being constrained by the requirement that they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

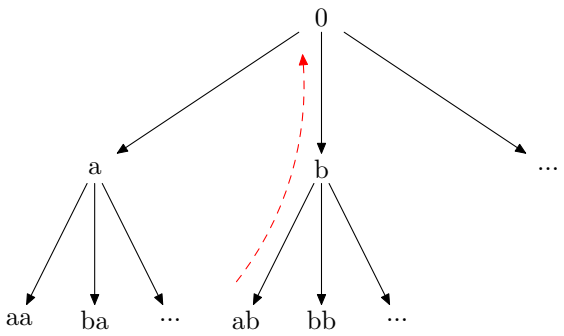


Figure 2.3: The hierarchy of contexts used in smoothing. The red dashed arrow indicates the escape path from 'ab' up to the root.

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,

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

where $\hat{p}_i(\cdot)$ 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 $\sum_i \lambda_i(\mathcal{C}) = 1$ 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,

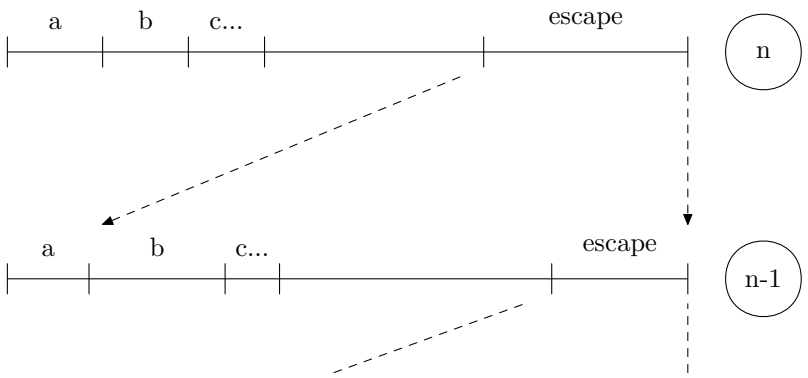


Figure 2.4: Escaping: The top line represents the full code space, and is divided according to the n^{th} level distribution, including some space allocated to the escape symbol. This space, which is represented by the second line is further divided according to the $(n-1)^{th}$ distribution, and in 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

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

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

where $q_n\left(\mathcal{C}\right)$ 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

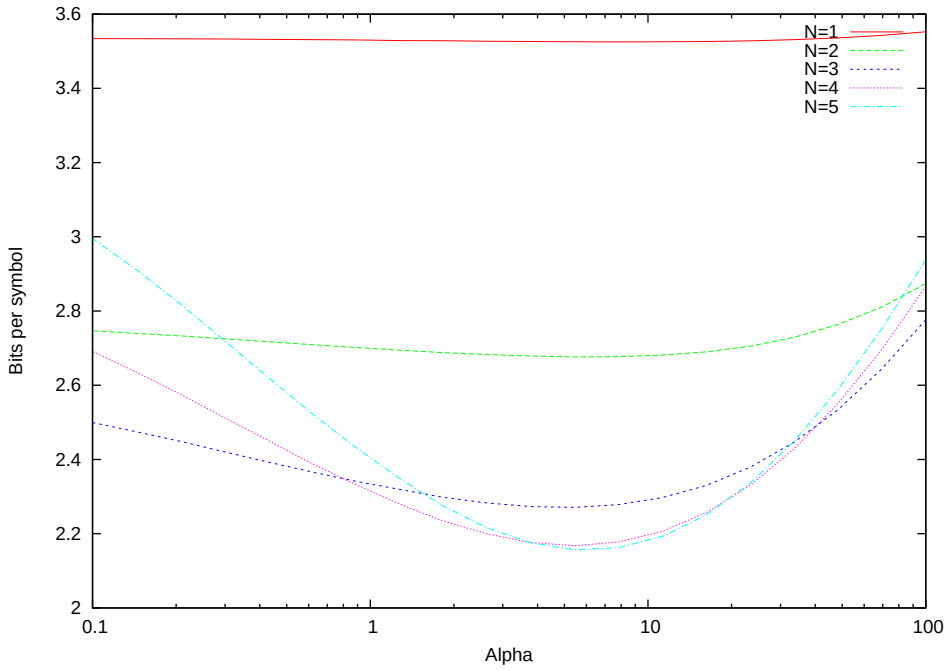


Figure 2.5: Experimental evaluation of the general PPM-A language model for values of N between 1 and 5, using an 100kB extract from `alice29.txt` for training and a separate 10kB extract from the same text for testing. For $N = 2$ the minimum information rate was achieved at $\alpha = 6.50$, giving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

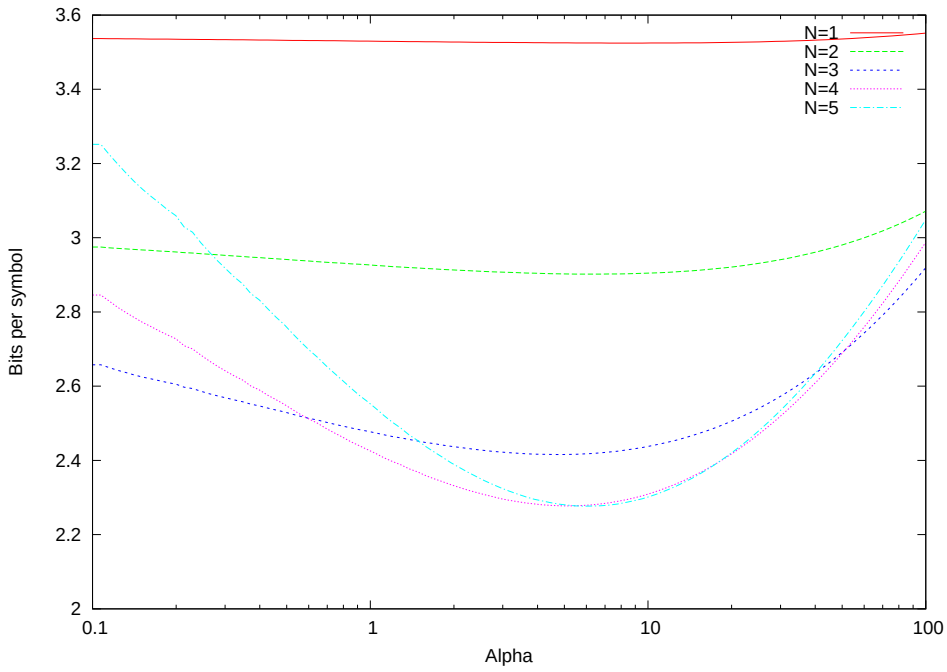


Figure 2.6: Experimental evaluation of the general PPM-A language model for values of N between 1 and 5, using an 100kB extract from the Enron corpus for training and a separate 10kB extract from the same text for testing. For $N = 2$ the minimum information rate was achieved at $\alpha = 6.31$, giving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

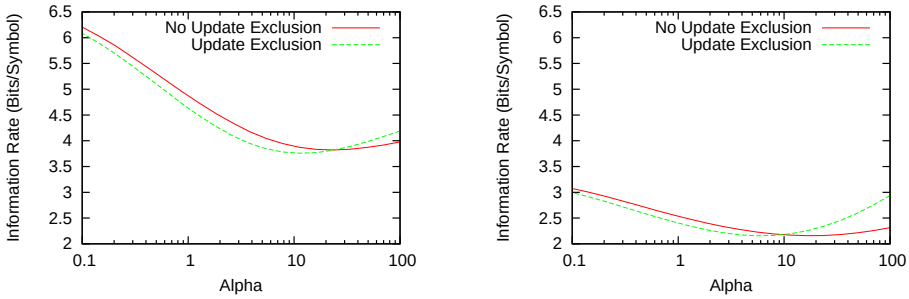


Figure 2.7: Effect of update exclusion. Compression rate as a function of α for training on 1kB and testing on 1kB (left) and for training on 100kB and testing on 10kB (right) using `alice29.txt`.

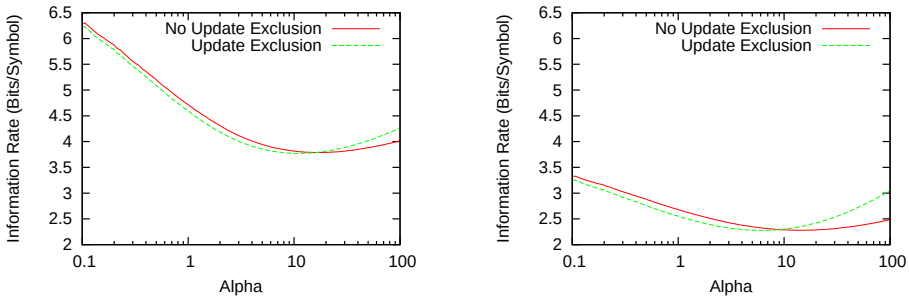


Figure 2.8: Effect of update exclusion. Compression rate as a function of α for training on 1kB and testing on 1kB (left) and for training on 100kB and testing on 10kB (right) using the Enron corpus.

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,

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

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

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

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

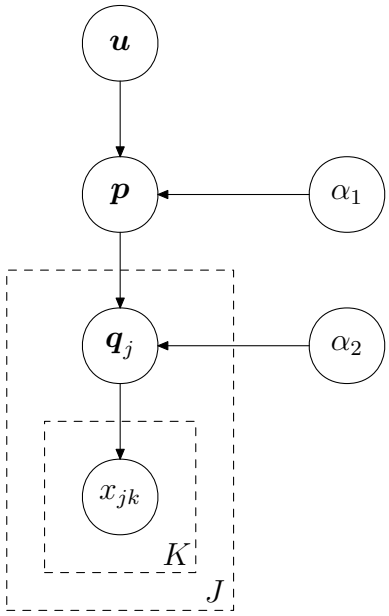


Figure 2.9: Bayesian network representing the hierarchical Dirichlet distribution. $\boldsymbol{u}$ is a base distribution, considered to be a parameter of the model, which is used to generate $\boldsymbol{p}$ from a Dirichlet distribution. This value is in turn used to generate a set of probability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

$$\boldsymbol{p} \sim \text{Dirichlet}\left(\alpha_1 \boldsymbol{u}\right) \tag{2.44}$$

$$\boldsymbol{q}_j \sim \text{Dirichlet}\left(\alpha_2 \boldsymbol{p}\right) \tag{2.45}$$

$$\boldsymbol{q}_{jk} \sim \text{Dirichlet}\left(\alpha_3 \boldsymbol{q}_j\right) \tag{2.46}$$

$$x_{jkl} \sim \text{Categorical}\left(\boldsymbol{q}_{jk}\right) \tag{2.47}$$

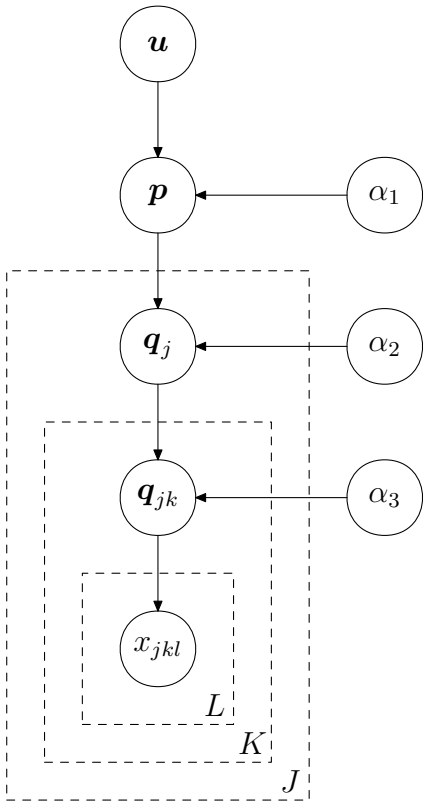


Figure 2.10: Bayesian network representing the three level hierarchical Dirichlet distribution.

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*,

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

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,

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

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

$$\begin{aligned}\Pr(x_{M+1} = j \mid x_1, \ldots, x_M) &= \frac{\Pr(x_1, \ldots, x_{M+1})}{\Pr(x_1, \ldots, x_M)} && (2.55)\\ &= \frac{\alpha u_j + m_j}{\alpha + M} && (2.56)\\ &= \frac{M}{\alpha + M} \cdot \frac{m_j}{M} + \frac{\alpha}{\alpha + M} \cdot u_j && (2.57)\end{aligned}$$

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 $\bar{m}_n(t \mid \mathcal{C})$ be the counts used when making predictions by the n^{th} level oracle corresponding to context $\mathcal{C}$, and $\bar{M}_n(\mathcal{C})$ 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.

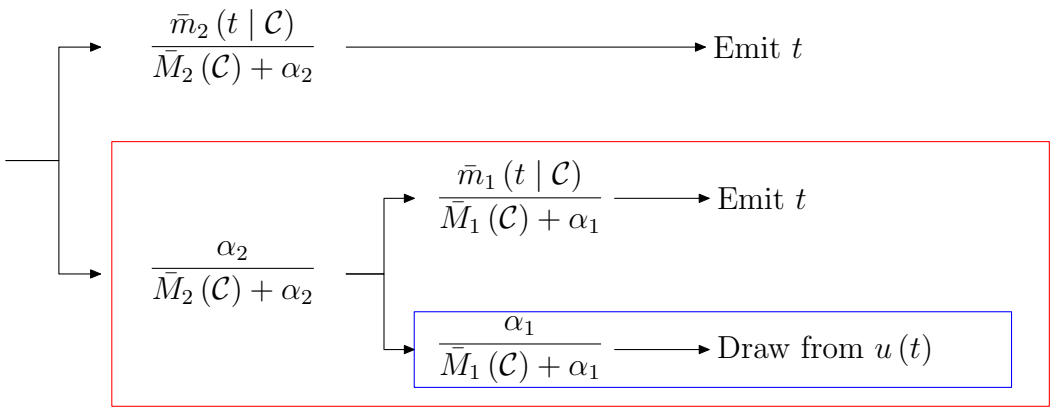


Figure 2.11: A schematic representation of the hierarchical Dirichlet distribution viewed in terms of a oracles. A red box is used to outline the mid level oracle, and a blue box indicates the top-level oracle, which in this case always draws from the distribution $u(t)$.

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

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

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

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

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

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

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

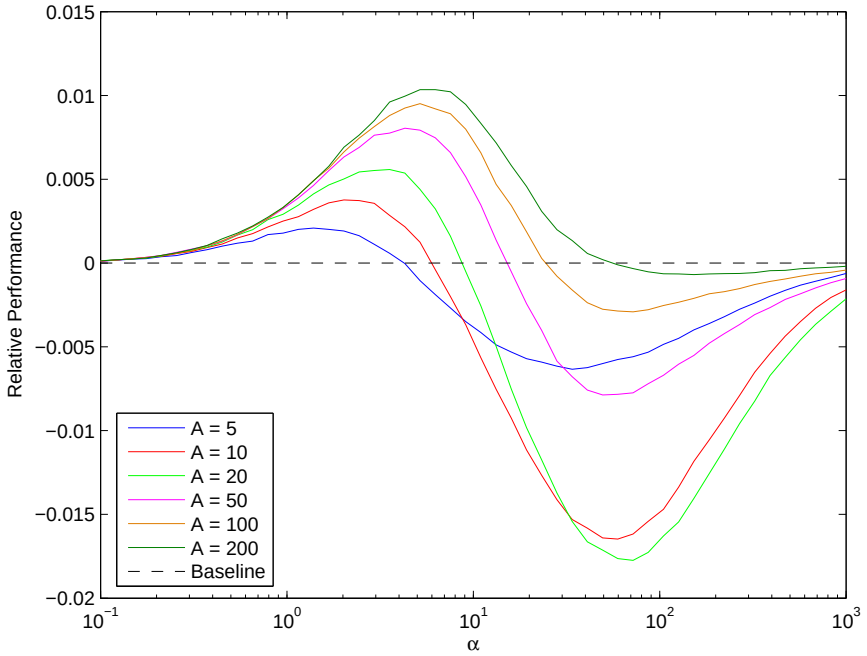


Figure 2.12: Relative performance of the two approximations as a function of alphabet size, A, and the Dirichlet parameter, α, under sequential generation. Results shown are Monte Carlo approximations using 10000 samples of length 1000. Positive values indicate better performance for the minimal oracle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

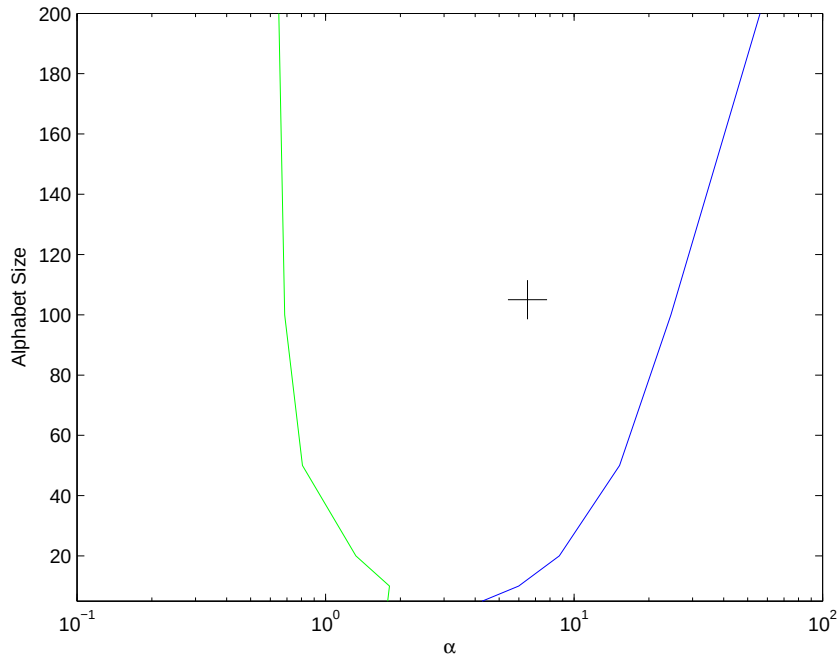


Figure 2.13: Boundary values of α, below which the minimal oracle approximation gives better performance. Curves are for sequential generation (blue) and random contexts (green). The black cross indicates the optimal value of $\alpha = 6.5$ found in Section 2.4 for bi-gram generalised PPM-A, which was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

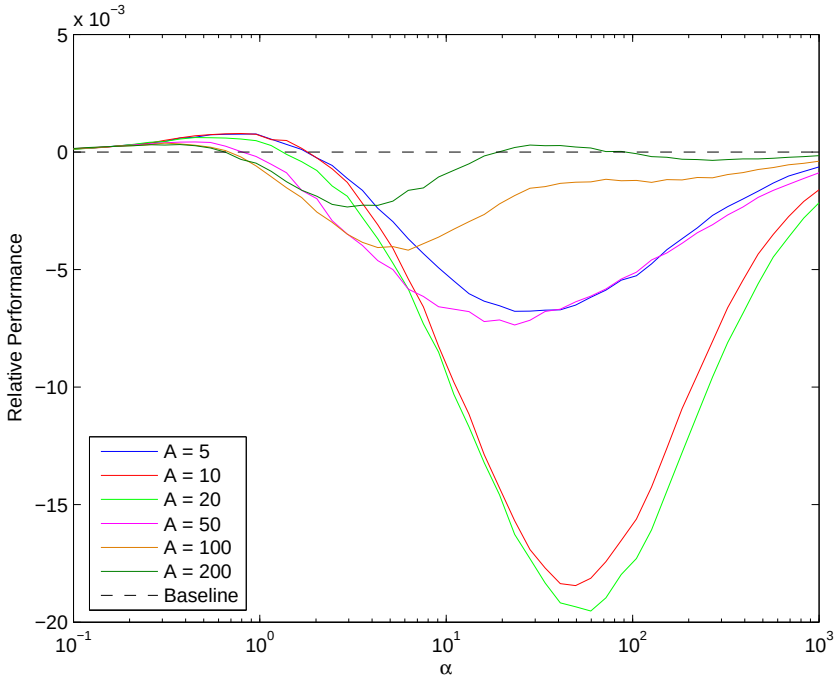


Figure 2.14: Relative performance of the two approximations as a function of alphabet size, A, and the Dirichlet parameter, α, when contexts are drawn at random from a uniform distribution. Results shown are Monte Carlo approximations using 10,000 samples of length 1,000. Positive values indicate better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

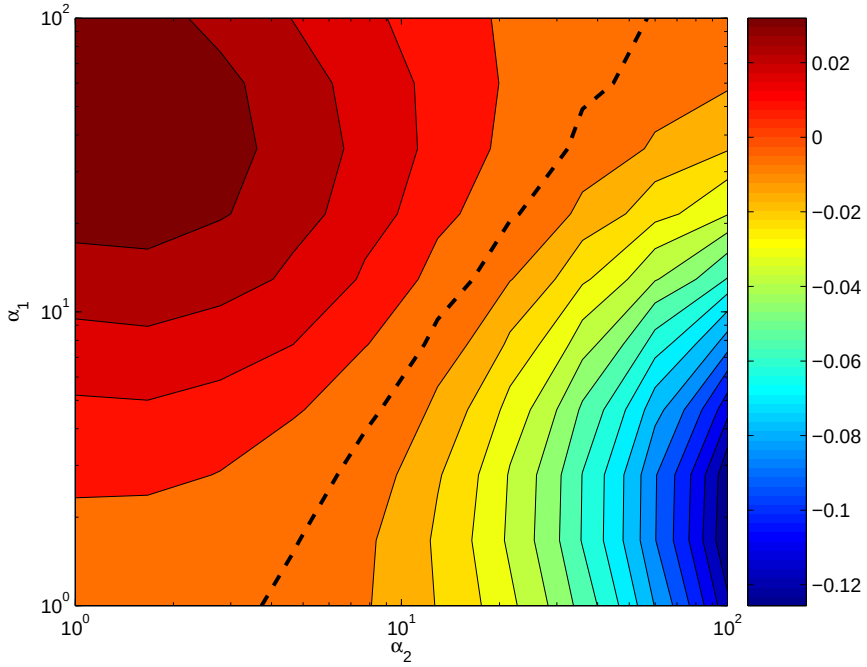


Figure 2.15: Contour plot showing the relative performance of the two approximations as α_1 and α_2 are varied independently. The plots are shown for 10,000 Monte Carlo samples of length 1,000, and an alphabet size of 100. The zero contour is highlighted as a dashed line. To the left of this contour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

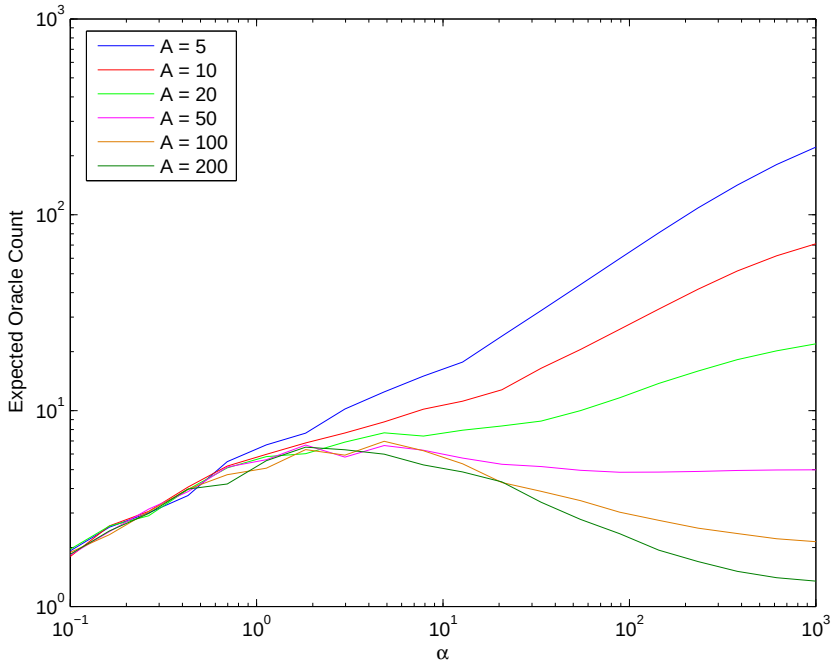


Figure 2.16: Mean number of oracle enquires per group as a function of the alphabet size and α. Results are mean values over ten repetitions, for data sets of length 10,000.

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.

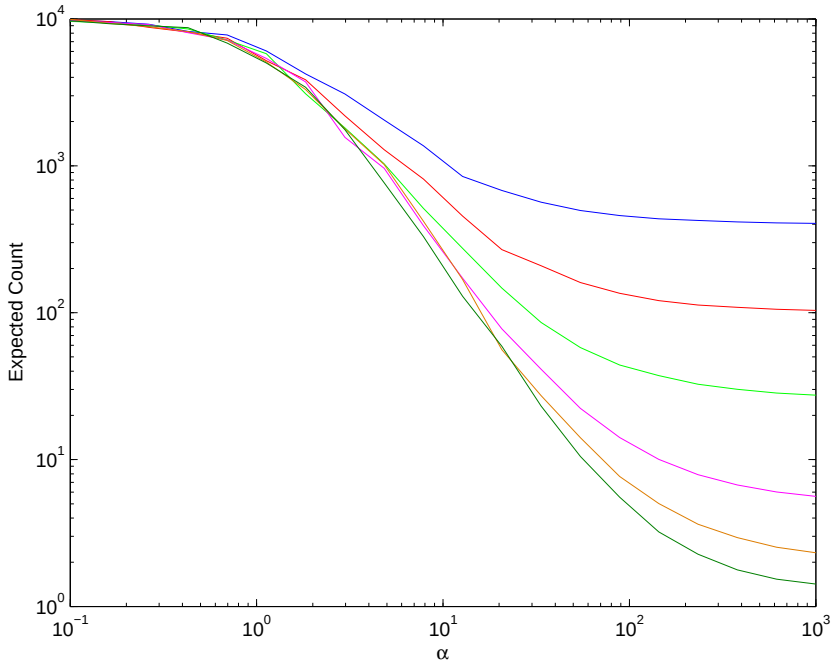


Figure 2.17: Mean number of entries per group as a function of the alphabet size and α. Results are mean values over ten repetitions, for data sets of length 10,000.

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

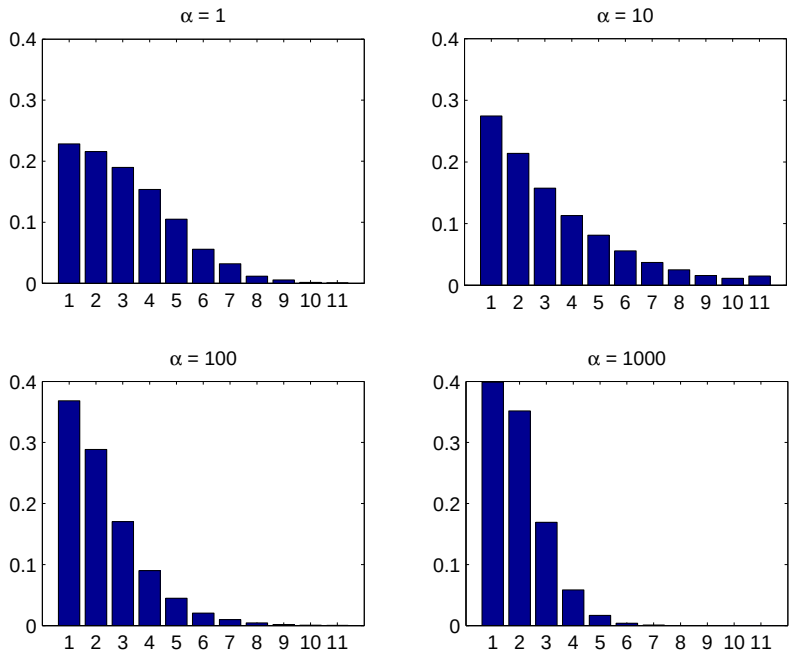


Figure 2.18: Bar charts showing the distribution of group counts for $\alpha = 1$, 10, 100 and 1000. Data is shown for an alphabet size of 105 and a data length of 10,000.

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

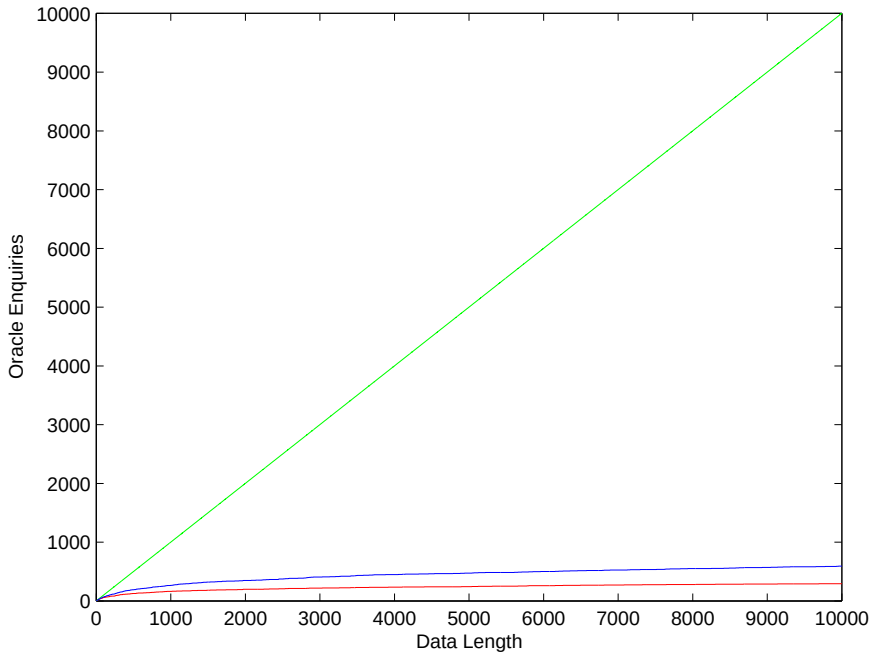


Figure 2.19: Number of oracle enquiries as a function of the number of data points generated. The blue curve shows the evolution of a typical sample from the prior distribution, whereas the red and green curves show the minimal and maximal oracle assumptions respectively.

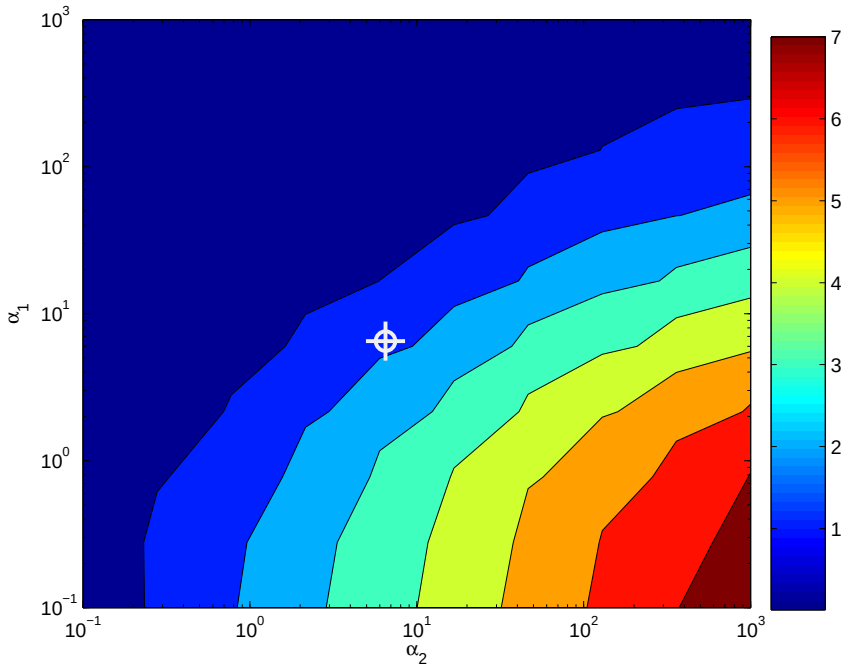
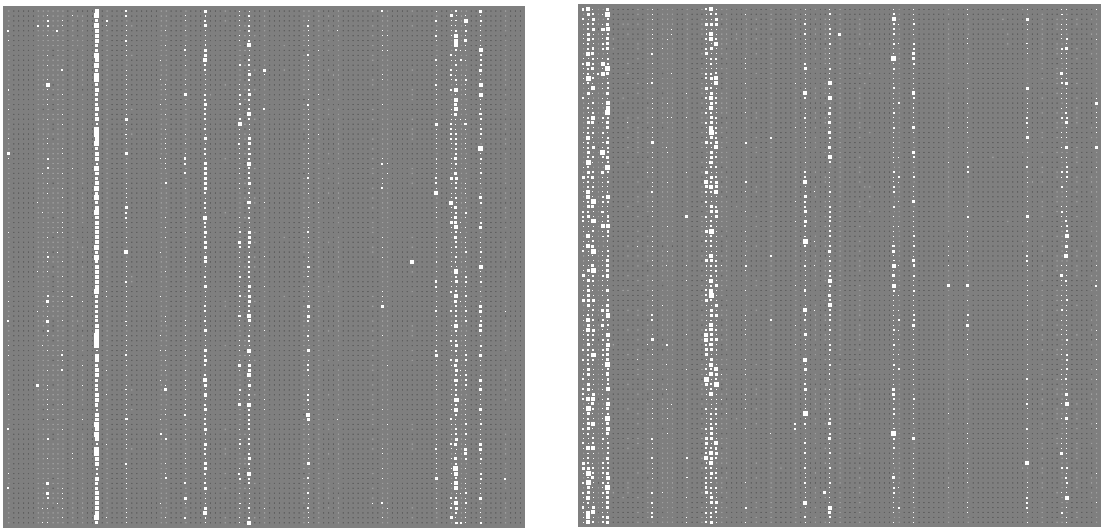
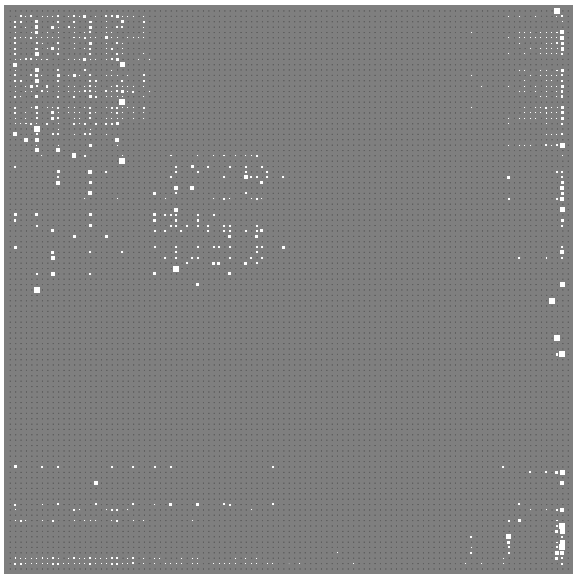


Figure 2.20: Mean number of groups (shown as logarithmic values) as a function of both α parameters for fixed alphabet size of 105. $\alpha_1 = \alpha_2 = 6.5$, the experimentally found values for generalised PPM-A, is indicated by a cross-hair.



(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 $\boldsymbol{z}^{\backslash i}$ 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 $z_i \sim P\left(z_i \mid \boldsymbol{z}^{\backslash i}, \boldsymbol{t}\right)$.
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

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

where $n_t(z_i)$ is the number of data points associated with oracle enquiry z_i, $n_z(t_i)$ 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 $\alpha = 6.5$

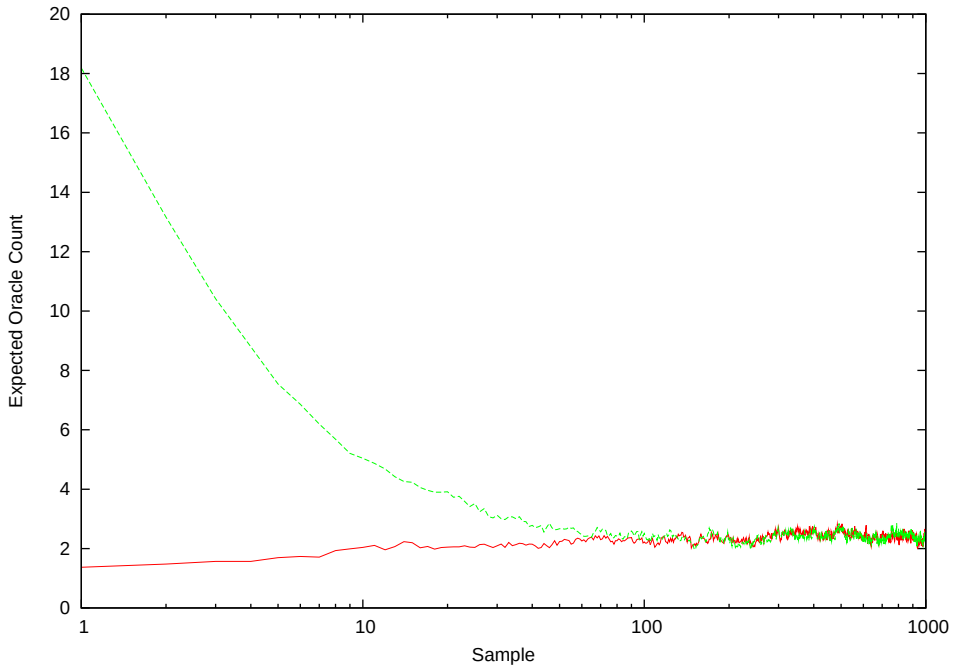


Figure 2.22: Samples from the posterior distribution, starting from both the minimal and maximal oracle assumptions. The average value over 800 iterations, allowing 200 iterations for burn in is 2.41 ± 0.01 in both cases.

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

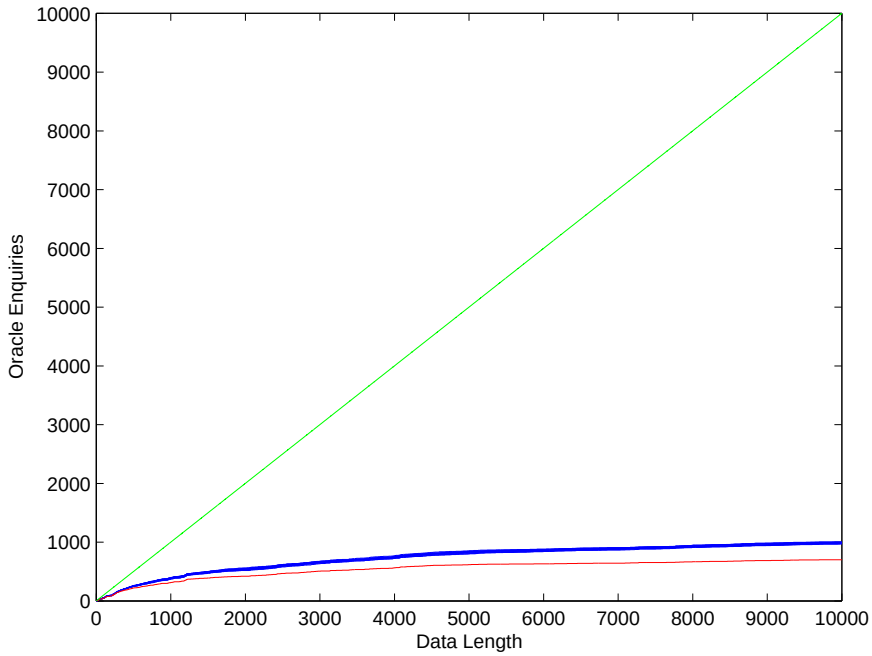


Figure 2.23: Number of oracle enquiries as a function of the number of data points generated. The blue curve shows the evolution of ten samples from the posterior distribution, whereas the red and green curves show the minimal and maximal oracle assumptions respectively.

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.

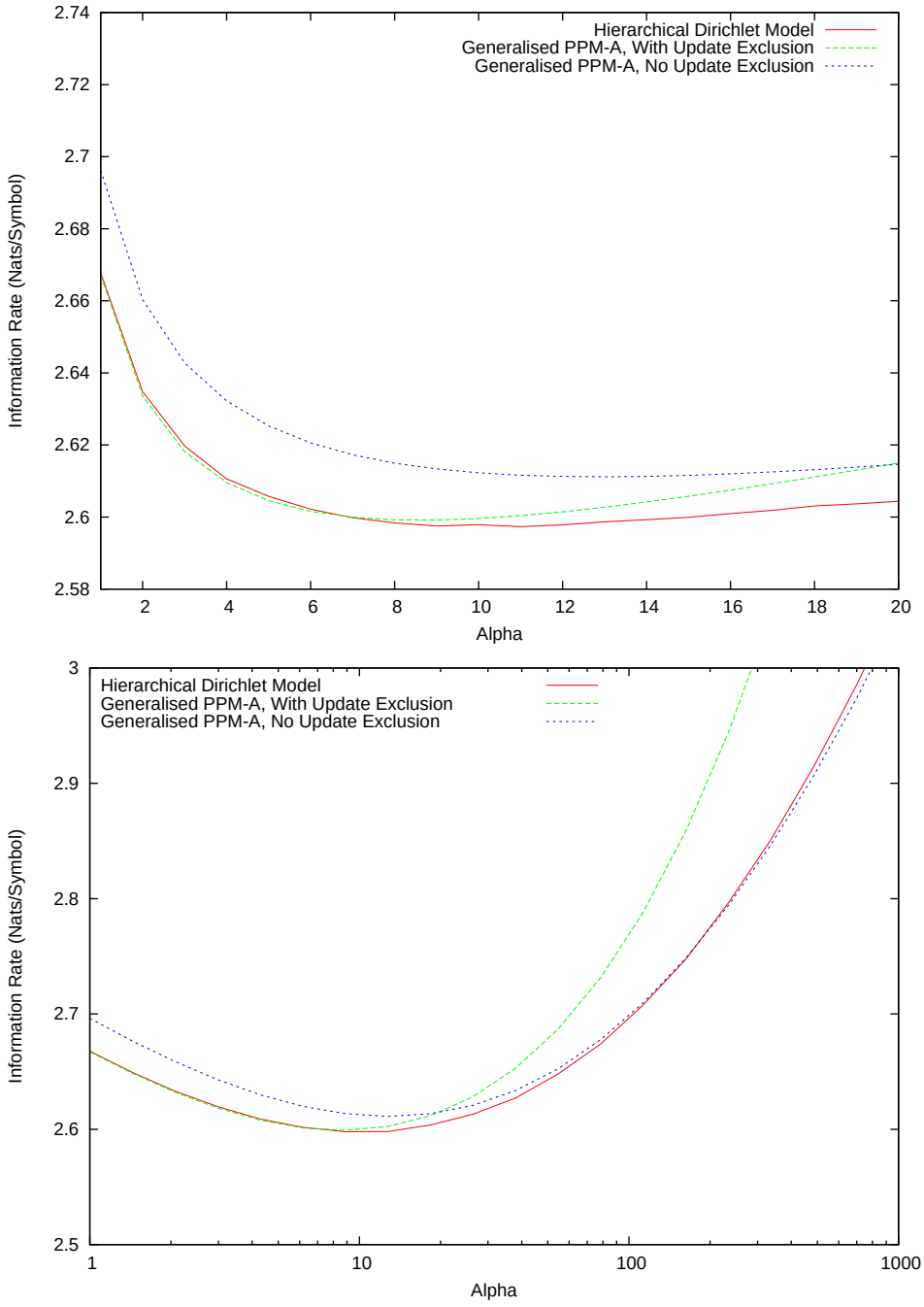


Figure 2.24: Comparison of the full hierarchical model to generalised PPM-A, using Monte Carlo simulation. Top: Detail of the region close to optimal performance. Bottom: Variation over a wider range of α, shown on a logarithmic scale.

Chapter 3

Word-Based Language Modelling

3.1 Introduction

The hierarchical Dirichlet language model used in Chapter 2 makes few assumptions about the structure of the text being modelled. Instead, information is obtained almost entirely through a process of learning from example data. The fact that a model requiring very little hand coded knowledge can be used successfully is of course of great interest, but in practice it may be desirable to encode existing knowledge into the model before any learning has taken place. One of the more notable pieces of structure in many languages is the division of the text stream into a sequence of interspersed word and non-word tokens [9], and one can imagine that this knowledge might help the predictive process. This division can be done deterministically using a few simple rules.

The use of word boundary information in letter based language modelling is not new — a variety of approaches have been used in the past [5] [33] [61]. This chapter has two goals: firstly to demonstrate how models making use of words-level information may be included in the theoretical framework developed in Chapter 2, and secondly to extend the word based approach to allow for the possibility of making use of a dictionary, even if it is not accompanied by information concerning the frequency of usage of the terms.

3.2 Theoretical development

A simple way of including word-level information is to use any of the standard language models described in Chapter 2, but using whole words as the primitive tokens, rather than individual letters. Such a model can be used to make symbol-level predictions through a simple summation. To see how this is done, consider the context, which can be expressed in terms of previously observed whole words, as well as symbols seen so far in the current word which will be referred to as $\hat{w}$. For example,

$$\underbrace{\text{some}}_{w_{M-3}}\ \underbrace{\textit{space}}_{w_{M-2}}\ \underbrace{\text{example}}_{w_{M-1}}\ \underbrace{\textit{space}}_{w_{M}}\ \underbrace{\text{te...}}_{\hat{w}}$$

The probability of the next symbol-level term being t is:

$$\Pr\left(t \mid w_1, \ldots, w_M, \hat{w}\right) = \frac{\sum_{w_{M+1} \in \mathcal{W}(t)} \Pr\left(w_{M+1} \mid w_1, \ldots, w_M\right)}{\sum_{w_{M+1} \in \mathcal{W}'} \Pr\left(w_{M+1} \mid w_1, \ldots, w_M\right)} \tag{3.1}$$

where $\mathcal{W}(t)$ is the set of all words with prefix $\hat{w}t$ and $\mathcal{W}'$ is the set of words with prefix $\hat{w}$. For example, if $\hat{w}$ is 'te', and t is 'x', $\mathcal{W}(t)$ contains all words starting 'tex...', whereas $\mathcal{W}'$ contains all words staring 'te...'.

A significant drawback with this approach using any of the language models defined so far is the need for a finite vocabulary to be defined in advance. This restriction neglects the fact that vocabularies in natural languages are essentially infinite due to the creation of new words, proper nouns and so on. Furthermore, in many applications it is not desirable to assign zero probability to any string, even if it is not a sequence of valid words. For example, in arithmetic compression documents containing mis-spelt words must still be compressible.

3.2.1 Infinite Dirichlet models

These problems can be avoided by using an infinite model. The Dirichlet process [27], which was briefly mentioned in Section 2.8, is a generalisation of the Dirichlet distribution which is suitable for this application. Whereas the Dirichlet distribution is a prior over finite discrete probability distributions, the Dirichlet process provides a prior over probability distributions defined on a continuous space. Draws from the Dirichlet process are purely atomic with probability one, but distribute probability mass over an infinite set of points.

The use of a continuous space is appropriate for the present application as it is possible to associate the infinite set of potential words with a set of points within this space. Using a prior which resulted in continuous distributions over this space would result in a probability of zero of the same word being drawn more than once, which would not result in a useful language model. The atomic distributions provided by the Dirichlet process avoid this problem.

To give a more formal definition of the Dirichlet process, let $\mathcal{S}$ be a continuous probability space, and let $\alpha\mathcal{U}$ be a measure defined on that space. As before, the convention is used that $\mathcal{U}$ is normalised, and α is a positive scalar. Let $U(\cdot)$ be defined according to the convention

$$U(P) = \int_P \mathrm{d}\mathcal{U} \tag{3.2}$$

with an equivalent definition for other measures. For any countable partition of $\mathcal{S}$, represented $\mathcal{P}$, with parts $\{P_i\}$, if $\mathcal{X}$ is a draw from a Dirichlet process with parameter $\alpha\mathcal{U}$ then $(X(P_1), X(P_2), \ldots)$ has a Dirichlet distribution with parameter $(\alpha U(P_1), \alpha U(P_2), \ldots)$.

The formal definition does not lend itself directly to the present application. There are however a number of alternative constructions which are more use in practice, including the stick-breaking construction [84] and the Chinese restaurant process [11]. The most relevant of these constructions is the Chinese restaurant process, which can be considered as the infinte generalisation of the Pòlya urn sampling scheme described in Section 2.5.4. The sampling

procedure is essentially the same in the two cases, except that rather than escaping to a uniform distribution over a discrete set, the oracle draws from the base distribution $\mathcal{U}$. In other words, the predictive distribution becomes

$$\mathcal{P}\left(x_{M+1} \mid x_1, \ldots, x_M\right) = \frac{\sum_{i=1}^{M} \delta_{x_i} + \alpha\mathcal{U}}{M + \alpha U\left(\mathcal{S}\right)} \tag{3.3}$$

where δ_x represents a unit measure concentrated at position x. Note that even if $\mathcal{U}$ is continuous, this construction makes it clear that the distribution $\mathcal{X}$ will be purely atomic. However, as will be explained below, $\mathcal{U}$ will always be discrete in natural language applications.

Note that the Dirichlet process in itself does not specify a particular mapping from samples from the domain of $\mathcal{U}$ onto the set of possible words. To achieve this a spelling model is required. When a new sample from $\mathcal{U}$ is generated, a new word must be generated from the spelling model and associated with that sample. Further details will be given in the following section.

The Dirichlet process may be arranged in a hierarchy in the same way as the Dirichlet distribution [92], resulting in a model which is similar to that described in the previous chapter, but over an infinite vocabulary. The oracle framework is equally applicable in this case, and was used as the basis of sampling schemes in [4] and [92], the latter case referring to the oracle scheme as the *Chinese restaurant franchise*. In practice, the generative process only differs from the hierarchical Dirichlet distribution when oracle enquiries are made right up to the top level of the hierarchy. In this case, a new sample is always drawn from the spelling model, rather than from a uniform distribution over a pre-defined set of tokens.

In the following sections, the same approximation will be made as was used to obtain generalised PPM-A from the hierarchical Dirichlet model. The minimal oracle assumption will be used, resulting in update exclusion as before, with the only significant difference being in the case when escaping occurs to the top level, in which case an infinite distribution is used as described in the next section.

3.2.2 Top level priors

As with the finite case, a top level prior distribution is required. The set of all possible finite length strings is countably infinite[1], so it is not possible to use a uniform distribution. One possible choice is simply the hierarchical Dirichlet language model as defined in Chapter 2, with a special symbol indicating that the end of the word has been reached. In the absence of any training data, this will simply return an exponential distribution over word lengths, with all strings of the same length receiving the same probability mass. As observations are made, the model will learn to predict plausible words.

Care must be taken as the statistics of the spellings of word and non-word tokens are likely to be very different. For this reason, in practice two models are used, one for each

[1]There are a finite number of strings of any given length, so consider enumerating all strings of length 1, followed by all strings of length 2 and so on...

	A	B	AB
Pre-trained spelling model	Yes	No	Yes
Interpolation	No	Yes	Yes

Table 3.1: Summary of the three dictionary methods.

type of token. As splitting of the stream into tokens is deterministic, and word and non-word symbols always alternate, choosing which model to use for predictions is a straightforward process. As before, the minimal oracle approximation will be used in the symbol-level model to give generalised PPM-A.

3.2.3 Using a word list

In some cases a word list will be available, but without frequency information. In this case it is not appropriate to use the word list as a training text for the full language model, as the model be trained as if all words are equally likely. It is however desirable to make use of the information somehow in order to obtain a model which will prefer valid to invalid words even if it is not possible to say which valid word is more probable.

One way in which this information could be used is in the spelling model which is used as part of the top level prior. Under the minimal oracle assumption, this distribution is only used when a previously unseen word is generated, so it is appropriate for this to reflect the letter level statistics of the language when a uniform distribution over possible words is considered, rather than when words are weighted by their frequency in typical texts. This approach can be implemented by using a letter level PPM-A spelling model, which is trained in advance on the word list.

An alternative approach is not use the same letter level spelling model, but not to train it in advance. Instead, predictions can be combined with those made by a separate model which is trained on the word list. The particular implementation considered below again makes use of a PPM-A spelling model, but interpolates predictions with a uniform distribution over all words in the word list.

Finally, the two methods described above are not mutually exclusive. It is possible to use a pre-trained spelling model as well as interpolating the predictions with a distribution over words. All three of these methods are evaluated in the remainder of this chapter, and are referred to as ‘Dictionary A’, ‘B’ and ‘AB’ respectively. These methods are summarised in Table 3.1.

3.2.4 Update exclusion

At first glance, it might seem appropriate to update the counts in the symbol-level model after every word-level token. However, under the minimal oracle assumption this model will be used only when a completely new word is seen. To be consistent it is therefore necessary to update the counts only in these cases. This behaviour will be tested empirically below.

3.3 Implementation details

As in (2.25), the model can be written as a linear combination of predictions made at different orders,

$$\Pr(w \mid \mathcal{C}) = \sum_{i=0}^{n} \lambda_i(\mathcal{C}) P_i(w \mid \mathcal{C}) \tag{3.4}$$

As the mixing coefficients don't depend on the term being predicted, summation over sets of words, as required in equation (3.1) can be done separately for each component of the sum,

$$\begin{aligned}
\sum_w \Pr(w \mid \mathcal{C}) &= \sum_w \sum_{i=0}^{n} \lambda_i(\mathcal{C}) P_i(w \mid \mathcal{C}) && (3.5)\\
&= \sum_{i=0}^{n} \lambda_i(\mathcal{C}) \sum_w P_i(w \mid \mathcal{C}) && (3.6)\\
&= \lambda_0(\mathcal{C}) \sum_w P_0(w \mid \mathcal{C}) + \sum_{i=1}^{n} \lambda_i(\mathcal{C}) \frac{1}{M_i(\mathcal{C})} \sum_w m_i(w \mid \mathcal{C}) && (3.7)
\end{aligned}$$

The summations can be computed in advance, and stored as total counts for each prefix in each context. This information is efficiently stored using a trie structure similar to that illustrated in Figure 3.1. The primary nodes of the trie are words, with the children representing possible continuations of the text up to some specified maximum depth. However, word nodes also have children corresponding to individual letters, which contain total counts for all child words with a given prefix. The word node itself stores the total count for all children, which is used as the denominator when calculating the predictive distribution.

The top level prior makes predictions in much the same way. For the case of the simple hierarchical model the probability of the letters seen so far in the current word is cached, and is multiplied by the predictive distribution returned by the symbol-level model. For the 'B' model, a trie is constructed representing the dictionary, which is used to make predictions in a similar fashion.

Prediction is made in two stages. In the first stage, some probability will be allocated to the 'end of word' symbol. In the second stage, the context is updated to include the current prefix as a completed word, and predictions are made for the first symbol in the next word. This distribution is scaled to replace the mass previously allocated to the end of word symbol. In the second stage, some probability mass may be allocated to the end-of-word symbol. This corresponds to a zero length word being inserted, and predictions after this word will be the same obtained in the second stage. This mass is therefore simply removed and the predictions renormalised appropriately before being combined into the first stage results.

We used $\alpha = 6.5$ in both the word level and spelling models, as suggested by experiments in Chapter 2. The spelling model used a maximum order of 5, and the word model used a maximum order of 3. We made no attempt to distinguish, for example, between differently capitalised instances of the same word, so 'Text' and 'text' for example are treated as different

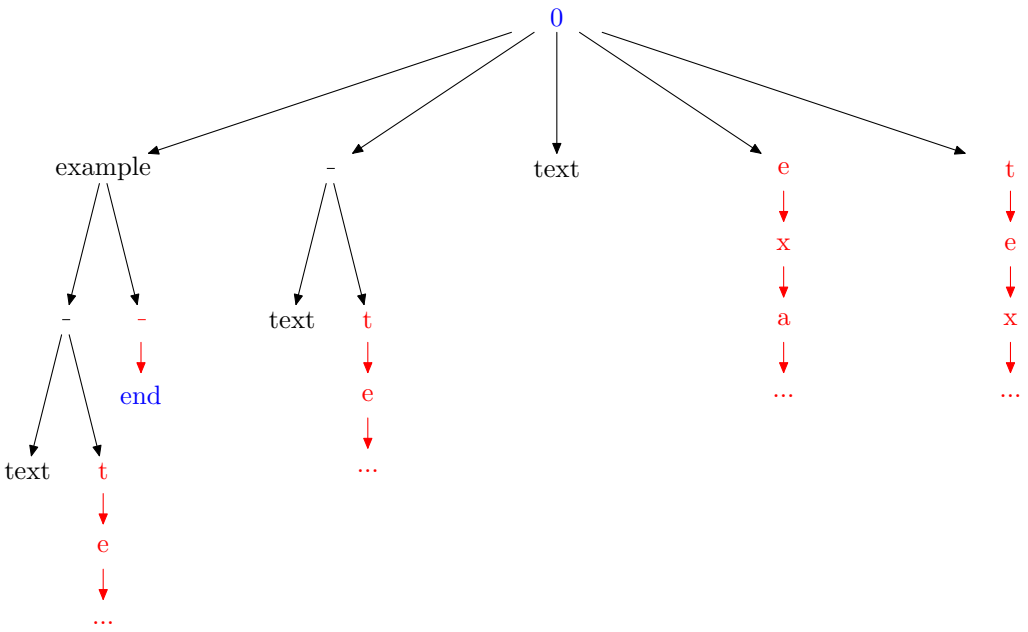


Figure 3.1: An illustration of the data structure used in the word-level model, shown storing the phrase '/example/_/text/'. Nodes representing words are shown in black, and those representing letters are shown in red. The special nodes, representing the root ('0') and the end of word marker ('end') are shown in blue.

words. The mixing coefficient in the case of dictionary methods B and AB was held at 0.5.

The word list used was taken from the ispell spell checking program [108], consisting of 96,030 words. The stream was divided into word and non-word tokens by assuming that letters as well as hyphens and apostrophes were always part of a word, and all other symbols were always part of a non-word token.

3.4 Results

We tested the model described above using the Enron corpus (See Appendix A), with 10kB of test text and non-adaptive testing. Results are given in Figure 3.2 showing predictive performance as a function of the size of training text, for both the case where no initial word list was used and the three dictionary methods described above. For purposes of comparison, the performance of generalised PPM-A with update exclusion operating only on the level of symbols is also shown.

The dictionary methods are able to significantly improve on the performance of the word-level model, particularly in the case of small quantities of training text. Out of the dictionary methods, method A was shown to perform better than method B. No improvement however was observed when the two dictionary methods were combined.

The word-level model incorporating a word list was able to outperform the symbol-level model only for training text sizes below approximately 8kB in length. This result is under-

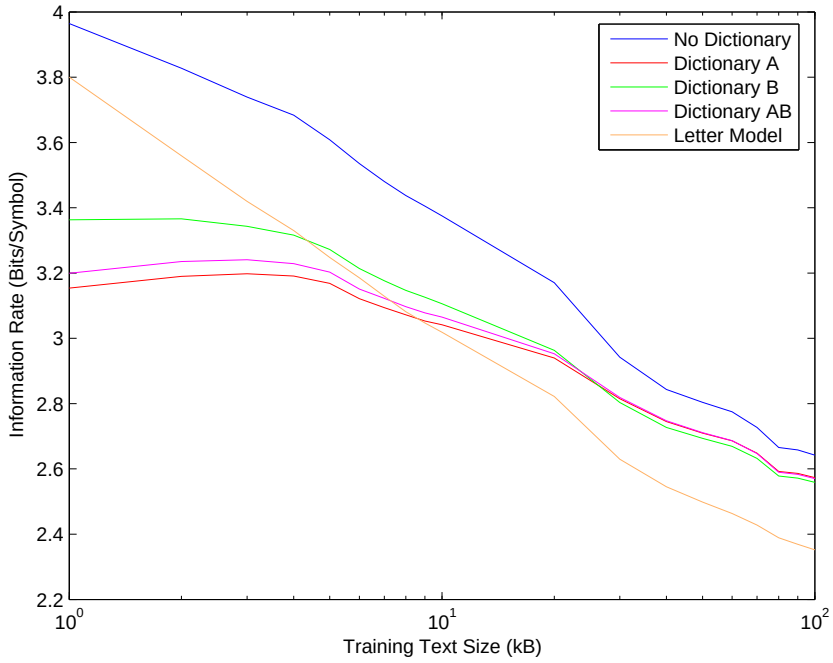


Figure 3.2: Comparison of experimental performance of the dictionary models using an in-domain test text.

standable, as for a given training text size there are fewer word tokens than letter tokens, which essentially gives the word-level model less information to learn from. Furthermore, although the word-level model does make use of a longer context it is more constrained in terms of what it can do with it. For example, the word-level model is unable to make predictions based on the suffix of the previous word (which is likely to contain information about its part-of-speech). It is nevertheless significant that an improvement is possible using the word model, and in some cases where no information on token frequency at all is available for training purposes this approach may be of use.

For comparison, Moffat [61] gives a figure of 2.79 bits per symbol for a 30,844 byte sample of English text, and 2.17 bits per symbol for a separate 131,521 byte text, using adaptive testing with no previous training text. The difference in methodology makes a direct comparison difficult, but these figures are roughly consistent with the results obtained here.

The previous tests were conducted in-domain: the training and test texts were taken from the same corpus and were therefore similar in style. We also examined how the performance changes if the test text is out-of-domain, shown in Figure 3.3. We performed this evaluation by substituting an extract from `alice29.txt` for the Enron test text (again, see Appendix A), but keeping the training text the same. In this case, the relative performance of the word-level model is weaker than in the case of in domain testing. Intuitively, this may be explained in terms of the degree of variability at the two levels; the word-level statistics of two

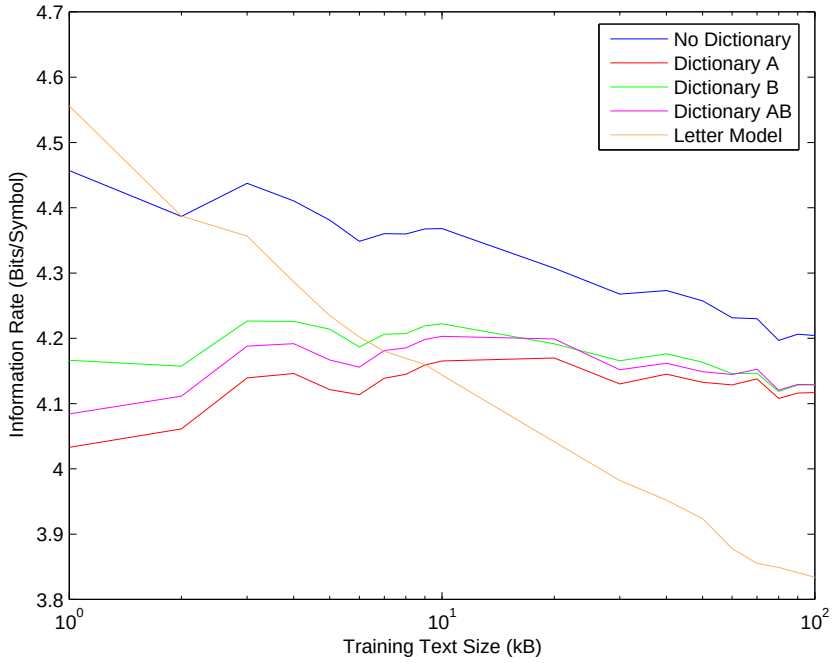


Figure 3.3: Comparison of experimental performance of the dictionary models using an out-of-domain test text.

texts in different domains are likely to be very difference, whereas the letter-level statistics are likely to be much more similar (assuming that the documents are in the same language). However, there is still a region of around 8kB where better performance is possible than the symbol-level model.

We also performed a final test to analyse the effect of the update exclusion principle discussed in Section 3.2.4. Using the word model without a dictionary and the in-domain test text, the performance was tested with and without update exclusion. The results are shown in Figure 3.4. An improvement in information rate of approximately 0.9% is observed when update exclusion is used.

3.5 Conclusions

This chapter has shown how the hierarchical Dirichlet framework may be extended to infinite vocabularies, and thus used to construct a language model which is able to make use of word boundary information. One application of such a model is in situations where a dictionary is known, but where frequency information is not available. It was shown that the word based model is able to use this information effectively to improve performance until sufficiently detailed statistics are learned from the text being modelled.

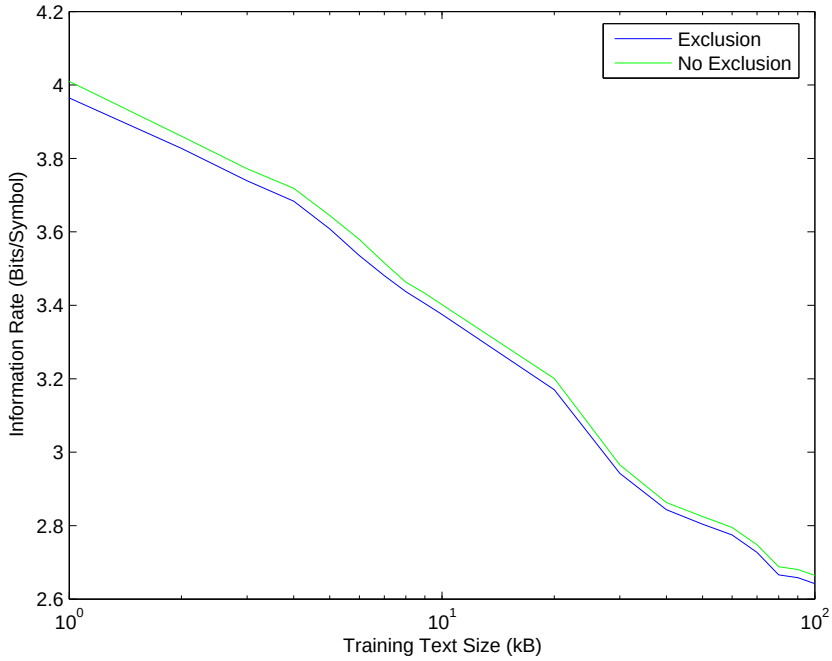


Figure 3.4: Effect of update exclusion when entering the symbol-level model.

Chapter 4

Text Retrieval

4.1 Introduction

Making large quantities of information available electronically is of limited use if it is not possible to obtain specific items easily. For this reason, this chapter considers text retrieval as a specific application of probabilistic language modelling. The primary focus is on *ad hoc* text retrieval. In this task, a collection of documents is provided, each of which consists of a string of text, possibly with some meta-information attached such as a title or a list of authors. Given a query, also consisting of a short string, the goal is to return an ordered set of documents such that documents which are most relevant to the query are returned first. Retrieval of richer documents, for example with layout information, or containing multimedia data, is considered to be outside of the scope of the problem. The notion of relevance is left vague, although it is usually compared against human judgements. Material in this section was presented in [22].

In this chapter, the symbol $\mathcal{C}$ will be used to represent the whole collection, which consists of documents $\{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots\}$. The symbol $\boldsymbol{q}$ will be used to represent the vector of terms present in the query.

4.2 Review of previous work

4.2.1 Vector space models

Vector space models [71] [57] were one of the earliest approaches to information retrieval. These models do not treat the problem probabilistically, but instead assign a vector to each document whose dimensionality is the size of the vocabulary,

$$\boldsymbol{d}_i = \sum_j d_{ij} \hat{\boldsymbol{t}}_j \tag{4.1}$$

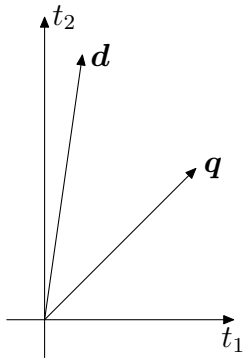


Figure 4.1: An illustration of the representation of a document, $\boldsymbol{d}$, and a query, $\boldsymbol{q}$, in the vector space model. In this example the vocabulary consists of just two terms, t_1 and t_2, which correspond to the to dimensions of the space. The components of each vector are the weights of these terms.

where $\left\{\hat{\boldsymbol{t}}_j\right\}$ is a set of basis vectors and d_{ij} is a *term weight* given to term j in document i. A corresponding vector is defined for the query,

$$\boldsymbol{q} = \sum_j q_j \hat{\boldsymbol{t}}_j \tag{4.2}$$

with q_j being the query term weight. This representation is illustrated in Figure 4.1.

The relevance is often defined in terms of the inner product of the normalised vectors,

$$R\left(\boldsymbol{d}_i, \boldsymbol{q}\right) = \frac{\boldsymbol{d}_i \cdot \boldsymbol{q}}{\left|\boldsymbol{d}_i\right| \left|\boldsymbol{q}\right|} \tag{4.3}$$

$$= \cos\theta_{\boldsymbol{qd}} \tag{4.4}$$

where $\theta_{\boldsymbol{qd}}$ is the angle between the vectors. It is common to use a form of *tf.idf* weighting, where *tf* stands for Term Frequency and *idf* stands for Inverse Document Frequency. In its simplest form this approach gives weights which are proportional to the number of times that each term appears in the corresponding document or query, and are inversely proportional to the number of documents in the collection containing the term. This is intuitively appealing, as a term which appears in many documents is less likely to be discriminative than one which only appears in a few. In practice a variety of non-linear transformations are used on the raw weights [57] [81].

4.2.2 Binary independence retrieval

Binary Independence Retrieval (BIR) express information retrieval in terms of probabilistic inference [72]. For each query there is assumed to be a set of relevant documents, with relevance being expressed in terms of the probability that a document is a member of this set. The original setup made use of a weight for each term appearing both in the document

and the query given by

$$w_i = \log \frac{p_i\,(1-q_i)}{q_i\,(1-p_i)} \tag{4.5}$$

where p_i and q_i are probabilities that a document chosen at random from the set of relevant and not relevant documents respectively contains term i. This expression is based on ratio of the odds of the terms appearing in the relevant and non-relevant sets respectively. Use of this model with estimates of these probabilities taken from the collection gives rise to the Robertson-Sparck Jones weight,

$$w_i^{RSJ} = \log \frac{(r_i + 0.5)\,(N - R - n_i + r_i + 0.5)}{(R - r_i + 0.5)\,(n_i - r_i + 0.5)} \tag{4.6}$$

where n_i is the number of documents containing term i, N is the total number of documents, r_i is the number of relevant documents containing the term, and R is the number of relevant documents. The additional values of 0.5 can be viewed as playing a role similar to smoothing using a Dirichlet prior as described in Section 2.3.2. Note that in order to make use of this weighting formula it is necessary to have estimates of the number of relevant documents and the associated word frequencies — a situation which is very rarely the case. At best we will have information concerning the relevance of a small number of documents based on user feedback, but in many cases none at all will be available. If this information is not known then a further simplification is often made,

$$w_i = \log \frac{N - n_i + 0.5}{n_i + 0.5} \tag{4.7}$$

which assumes that the relevant set is of negligible size compared to the whole collection. In this case, n_i can be approximated by the overall collection frequencies of terms. This function decreases monotonically with n_i/N. In other words, higher scores are assigned to terms which appear in few documents and are therefore highly discriminative, essentially embodying the principle of *idf* weighting.

Note that there is no term frequency component in (4.6). The weighting scheme is typically used in conjunction with other terms which reflect other aspects of the retrieval process. The most successful of these approaches is the Okapi BM-25 scheme [76]. The full score function in this case assigns a score of

$$s_i = \frac{(k_3 + 1)\,q_i}{k_3 + q_i} \cdot \frac{(k_1 + 1)\,f_i}{K + f_i} \cdot w_i^{RSJ} \tag{4.8}$$

to each term in the query, where q_i and f_i are the frequencies of term i in the query and document respectively, and

$$K = k_1\,(bL + (1 - b)) \tag{4.9}$$

with L being the overall length of the document. The relevance is the sum of these scores for

all terms in the query. Optionally an additional score of

$$k_2 n_q \frac{1-L}{1+L} \tag{4.10}$$

can be added for each document, providing normalisation of the document length, which is necessary to compensate for the fact that large documents are more likely to contain query terms 'by chance'. The quantities k_1, k_2, k_3 and b are parameters of the score function and must be set in advance, for example by optimisation over a training set of queries. A good summary of the development of these methods is provided in [75].

4.2.3 Language modelling approaches

A more recent approach to probabilistic retrieval makes use of language modelling [66]. In its original form, each of the documents in the collection is treated as being drawn from a separate language model. It is assumed that there is one relevant document, and that the query was generated from the same language model as that document. Bayes' rule can then be used to find the probability that each document is relevant. An alternative approach is to construct a language model from the query, and to rank documents according to the probability that they were generated by that language model. Further variations have considered the task to be equivalent to machine translation, where the 'query language' must be translated into the 'document language' or vice versa [10], or have used a measure of similarity such as the KL divergence [49].

As is the case with the language models described in Chapter 2, maximum likelihood estimates of the term distribution result in very poor performance, and it is generally found that smoothing is needed. Typically, smoothing is performed against a fixed base distribution, P^{base}, with the following two methods being common:

- **Linear Interpolation:** In this case, the maximum likelihood predictions for the document are simply mixed linearly with the base distribution

$$P(t) = \lambda P^{ML}(t) + (1-\lambda) P^{base}(t) \tag{4.11}$$

- **Dirichlet Smoothing:** This approach uses the predictive distribution under a Dirichlet prior where αP^{base}, for some positive constant α, is used as the base measure. The predictive distribution is therefore

$$P(t) = \frac{N}{N+\alpha} P^{ML}(t) + \frac{\alpha}{N+\alpha} P^{base}(t) \tag{4.12}$$

  where N is the total number of terms in the document.

The choice of the distribution with which to smooth is generally treated as external to the derivation of the model, and is often the maximum likelihood estimate of the distribution

for the collection as a whole. A notable exception is that used by the Twenty-One Consortium [31], which uses the normalised vector of document frequencies, although this is very much a heuristic choice. It is worth remarking that in the Dirichlet smoothing model, the choice of a prior distribution based on the data being modelled is not theoretically valid, so the model cannot be considered to be truly Bayesian.

4.2.4 Relevant sets

The methods described above differ in their approach to the relevant set. The BIR family of models are based on the assumption that there is set of relevant documents, the size of which is not specified in advance. The language modelling approach however often works under the assumption that there is precisely one document which is relevant to each query (although alternative interpretations are possible within this framework, for example in [47]).

Whether the assumption of a single relevant document is appropriate is a matter of debate, and quite probably depends on the nature of the application. Creation of language modelling strategies which do not make this assumption is certainly an area for future work in the field. It should also be noted that the evaluation method, to be discussed below, does assume multiple relevant documents, and therefore there is to some extent a conceptual gap between theory and practice for these approaches.

4.2.5 The probability ranking principle

How should information retrieval results be presented to the user? Assuming that relevance is treated as a binary property, so that a document is either relevant to a particular query or it isn't, the *probability ranking principle* [73] asserts that the optimum ordering is to present documents in decreasing order of probability of relevance. This result makes the assumption that relevances are independent. In other words, the probability of one document being relevant is not affected by the relevance of any other document. In practice the assumption of independence does not strictly hold, with the extreme case being duplicate documents whose relevance is fully correlated.

For the language modelling approach, the probabilistic interpretation of the relevance may be the probability of that document being the single item relevant to the query. In this case, it is possible to consider a cost function, c_i, which increases monotonically with the position of the relevant document in the returned list. The expected cost of a ranking is therefore

$$\langle c \rangle = \sum_i p_i \; c_{r_i} \tag{4.13}$$

where p_i is the probability that the i^{th} document is relevant and r_i is the position of document i in the returned list. This is minimised by returning the most probable documents first, with less probable documents being placed in positions which incur a higher cost.

4.2.6 Relevance feedback

Suppose that a retrieval has been performed using a query supplied by the user and resulting a set of documents which have been returned. Out of the returned documents, the user may indicate that some are relevant and some are not. This additional information can then be used to re-rank the collection, with the aim of improving the retrieval results. This approach is known as *relevance feedback*. A variation on this idea is *pseudorelevance feedback*, where the top few results of the initial query are assumed to be relevant when performing the re-ranking, avoiding the need for explicit user intervention.

The details of the approach vary from one retrieval algorithm to another, but in general the technique involves some combination of introducing additional query terms, and altering the weighting of exiting terms in the query. For example, in the vector space retrieval model, one approach is to move the query vector towards the centre of the relevance documents and away from the centre of those which are irrelevant:

$$\boldsymbol{q}_1 = a\boldsymbol{q} + b\sum_{i\in\mathcal{R}}\frac{\boldsymbol{d}_i}{|\mathcal{R}|} - c\sum_{j\in\mathcal{N}}\frac{\boldsymbol{d}_j}{|\mathcal{N}|} \tag{4.14}$$

where $\mathcal{R}$ and $\mathcal{N}$ are the sets of relevant and irrelevant documents respectively, and a, b and c are parameters. This approach is known as the Rocchio method [77]. Similar methods also exist for the BIR method [74].

When relevance feedback is used to expand the query one interpretation of the effect is that terms which are relevant to topics expressed in the query, but which are not contained in the query itself, are identified and included in the search. This approach is similar to Latent Dirichlet Allocation and other topic models described in Section 2.3.10 — one could imagine that both the query and the documents could be converted into a representation based on topics, and those representations could be used in place of the document terms themselves during information retrieval. Applications in information retrieval were indeed a significant motivation behind the development of topic models, and while they are not considered further in this thesis, it is likely that such models will play an important role in the future development of language modelling retrieval methods.

4.3 Whole collection models

The approaches described in Section 4.2.3 all treat the language models learned for each document as being independent. In this section an approach is considered which relaxes this requirement, permitting sharing of information between different documents.

Each document in the collection consists of a sequence of terms, denoted by $(t_1, t_2 \ldots)$. By assigning a label, y_i, to each term indicating the document from which it was taken, the whole collection can be viewed as a single set of (t, y) pairs. It is possible to define a probabilistic model over such pairs. To perform information retrieval, a label is associated with each term

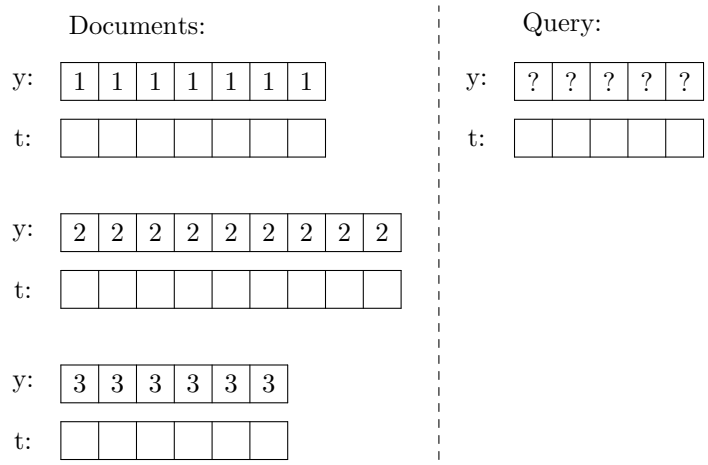


Figure 4.2: Illustration of the whole collection model. Associated with each term in the collection is a label indicating to which document it belongs. A label is also associated with each term in the query, which is viewed as an unobserved random variable. Relevance is defined as the probability that these labels will take the same value as those for a particular document.

in the query as well. It is assumed that the query (t, y) pairs were generated from the same model as those in the collection, with the constraint that the query labels must all take the same value, y_q. The relevance of document $\boldsymbol{d}$ to $\boldsymbol{q}$, $R(\boldsymbol{d}, \boldsymbol{q})$ can now be defined as the log probability that $y_q = y_d$, the label corresponding to $\boldsymbol{d}$. See Figure 4.2 for an illustration.

The symbols $\boldsymbol{t}_\mathcal{C}$ and $\boldsymbol{y}_\mathcal{C}$ will be used to denote vectors of the t and y values for all terms in the collection, and $\boldsymbol{t}_q$ denotes the vector of terms in the query. Predictions for y_q can be made using Bayes' rule,

$$\Pr(y_q \mid \boldsymbol{t}_q, \boldsymbol{y}_\mathcal{C}, \boldsymbol{t}_\mathcal{C}) \propto \Pr(\boldsymbol{t}_q \mid y_q, \boldsymbol{y}_\mathcal{C}, \boldsymbol{t}_\mathcal{C}) \cdot \Pr(y_q \mid \boldsymbol{y}_\mathcal{C}, \boldsymbol{t}_\mathcal{C}) \tag{4.15}$$

The prior, $\Pr(y_q \mid \boldsymbol{y}_\mathcal{C}, \boldsymbol{t}_\mathcal{C})$, is the probability that a given document is relevant in the absence of a query, and can be used to incorporate additional information. For example, an analysis of link structure in a hypertext collection using an algorithm such as PageRank [14] or hubs and authorities [42] can be used to estimate this probability. In this chapter however a uniform prior will be used.

It is possible to express the language modelling approach to information retrieval within this framework by using

$$\Pr\left(t_q^{(i)} \mid y_q, \boldsymbol{y}_\mathcal{C}, \boldsymbol{t}_\mathcal{C}\right) = P_{y_q}\left(t_q^{(i)}\right) \tag{4.16}$$

where $P_{y_d}(t)$ is the language model corresponding to document $\boldsymbol{d}$. The approach presented here can therefore be viewed as a generalisation of existing methods.

4.3.1 The model

The proposed model consists of a separate discrete distribution over terms for each document in the collection. The distribution corresponding to document j has parameter $\boldsymbol{q}_j$. Documents are constructed by making i.i.d. draws of terms from the corresponding distribution. The set of parameters for each of the discrete distributions is drawn from the hierarchical Dirichlet model as described in Section 2.5.1,

$$\begin{aligned} \boldsymbol{p} &\sim \text{Dirichlet}\left(\alpha_1 \boldsymbol{u}\right) && (4.17)\\ \boldsymbol{q}_j &\sim \text{Dirichlet}\left(\alpha_2 \boldsymbol{p}\right) && (4.18)\\ t_k &\sim \text{Categorical}\left(\boldsymbol{q}_{d_k}\right) && (4.19) \end{aligned}$$

This model is equivalent to that illustrated in Figure 2.9. The same approximation is used as in generalised PPM-A. Namely that when making predictions it is assumed that the oracle was asked precisely once for each term observed in each document. This means that the oracle counts will be exactly equal to the number of documents containing the corresponding term, or in other words the document frequency of the term. The model is further approximated by ignoring the effect on the predictive distribution of query terms that have already been seen when making predictions. This gives

$$\Pr\left(t \mid y\right) = \frac{1}{N_{y_d} + \alpha_2}\left(n_t\left(t, y\right) + \alpha_2 \,\hat{\mathrm{p}}\left(t\right)\right) \tag{4.20}$$

where

$$\hat{\mathrm{p}}\left(t\right) = \frac{n_d\left(t\right) + \alpha_1 / \left|\mathcal{V}\right|}{\sum_{t'} n_d\left(t'\right) + \alpha_1} \tag{4.21}$$

which can be interpreted as a modified document frequency term. $n_t\left(t, y\right)$ is the term frequency, or the number of times that term t appears in the document with label y and $n_d\left(t\right)$ is the document frequency, i.e. the number of documents containing t at least once. N_{y_d} is the length of document $\boldsymbol{d}$, and $\left|\mathcal{V}\right|$ is the total number of different terms in the collection.

The log probability of the whole query after a little rearrangement is

$$\begin{aligned} \log \Pr\left(\boldsymbol{t}_q \mid y_q\right) = &\sum_i \log\left(\frac{1}{N_{y_q} + \alpha_2}\right)\\ &+ \sum_i \log\left(1 + \frac{n_t\left(t_q^{(i)}, y_q\right)}{\alpha_2 \,\hat{\mathrm{p}}\left(t_q^{(i)}\right)}\right) + \sum_i \log\left(\alpha_2 \,\hat{\mathrm{p}}\left(t_q^{(i)}\right)\right) \end{aligned} \tag{4.22}$$

Note that the last term in this expression is not dependent on the query, and therefore may

be ignored for the purposes of ranking. The relevance can therefore be written as

$$R(\boldsymbol{d},\boldsymbol{q}) = \sum_i \log\left(1 + \frac{n_t\left(t_q^{(i)}, y_d\right)}{\alpha_2\,\hat{\mathrm{p}}\left(t_q^{(i)}\right)}\right) + N_q \log\left(\frac{1}{N_{y_d}+\alpha_2}\right) \tag{4.23}$$

where N_q is the length of the query and the sum runs over terms appearing in the query, such that terms appearing multiple times in the query contributing multiple times to the sum. The first term in this expression provides *tf.idf*-like query term weighting. The second term may be interpreted as providing global document length normalisation.

The hierarchical model is in many respects similar to Dirichlet smoothing, and therefore recovers the same document length normalisation term. The most important difference however is in the form of $\hat{\mathrm{p}}$ which arises *internally*. The only arbitrary choice is the top level prior, which is uniform. Conceptually this is a significant improvement over existing models, which as mentioned before either neglect document frequency information entirely, or introduce it in an *ad hoc* fashion.

The form of $\hat{\mathrm{p}}$ also solves a minor technical issue, namely the fact that both the maximum likelihood document and collection models assign zero probability to terms which have not been observed anywhere in the collection. Formally this results in a divergent score function, making it impossible to differentiate between documents for any query containing such a term. Of course, this can be solved by simply ignoring any terms which are absent in the collection, as they are not able to affect the results, but it is appealing to be able to produce a model which inherently avoids this problem.

4.3.2 Experimental evaluation

We tested the model on the *ad hoc* tasks from TREC-7 and TREC-8, as well as the Cranfield data set (see Appendix A). The TREC tasks each consist of a set of 50 queries complete with ground-truth relevance judgements against a standard set of documents. As specified for the respective TREC conferences, these tasks used all data on discs 4 and 5 except for the CR texts and with queries 351-400 and 401-450 respectively. The full query text consisting of the title, description and narrative was used in all cases (see Section 4.4.4 below).

The Cranfield data set is much smaller, and much more specialised, containing abstracts from technical papers on aeronautical engineering. In this case 225 queries are provided, again with ground-truth relevance judgements.

The only pre-processing that we did prior to indexing was basic stop-word removal (see Appendix B) and stemming using a Porter stemmer [67]. We performed the experiments using the LEMUR language modelling toolkit [64], which provides amongst other things a set of indexing routines as well as implementations of standard methods such as BM-25.

Precision and recall

Two metrics are commonly used to describe the effectiveness of information retrieval algorithms: precision and recall. Precision is defined by

$$\text{Precision} = \frac{\text{Number of relevant documents returned}}{\text{Number of documents returned}} \tag{4.24}$$

and recall by

$$\text{Recall} = \frac{\text{Number of relevant documents returned}}{\text{Total number of relevant documents}} \tag{4.25}$$

These quantities will clearly vary depending on the number of documents returned. Returning just one document will result in a high precision (assuming that document is relevant), but at a low recall. The converse will be true in the limit that the whole collection is returned. To indicate this trade off, precision–recall curves are used, showing how the two quantities vary with the number of documents returned. The performance can be further summarised by computing the average precision after each relevant document is returned

$$\text{Average precision} = \frac{1}{N} \sum_{n \in \mathcal{R}} \text{Precision}_n \tag{4.26}$$

Alternatively, one can consider the precision after a fixed number of documents have been returned. Many conventional search engines display ten documents per page, and it is likely that the user will not ask for a second page to be displayed, but will rather refine their search if a relevant document has not been found. Precision after ten documents is therefore the measure which was chosen in this work.

4.4 Results

4.4.1 Precision experiments

Average precision results for the various methods are shown in Figure 4.3, and precisions at 10 documents are shown in Figure 4.4. In all cases, α_1 was held fixed at 1,000 while α_2 was allowed to vary. Results for BM-25 (see Section 4.2.2) are shown too as horizontal dashed lines. The default values were used for the BM-25 parameters: $k_1 = 1.2$, $k_3 = 7$ and $b = 0.75$. No document length normalisation was performed, so the k_2 parameter is not required.

In all measures with the exception of average precision using the Cranfield data set, the hierarchical method gives the highest performance. In all of the experiments using the TREC data set the hierarchical method was able to out-perform BM-25, both in terms of the average precision and the precision at ten documents.

Cranfield:
1400 docs
225 queries

TREC 7:
528,155 docs
50 queries

TREC 8:
528,155 docs
50 queries

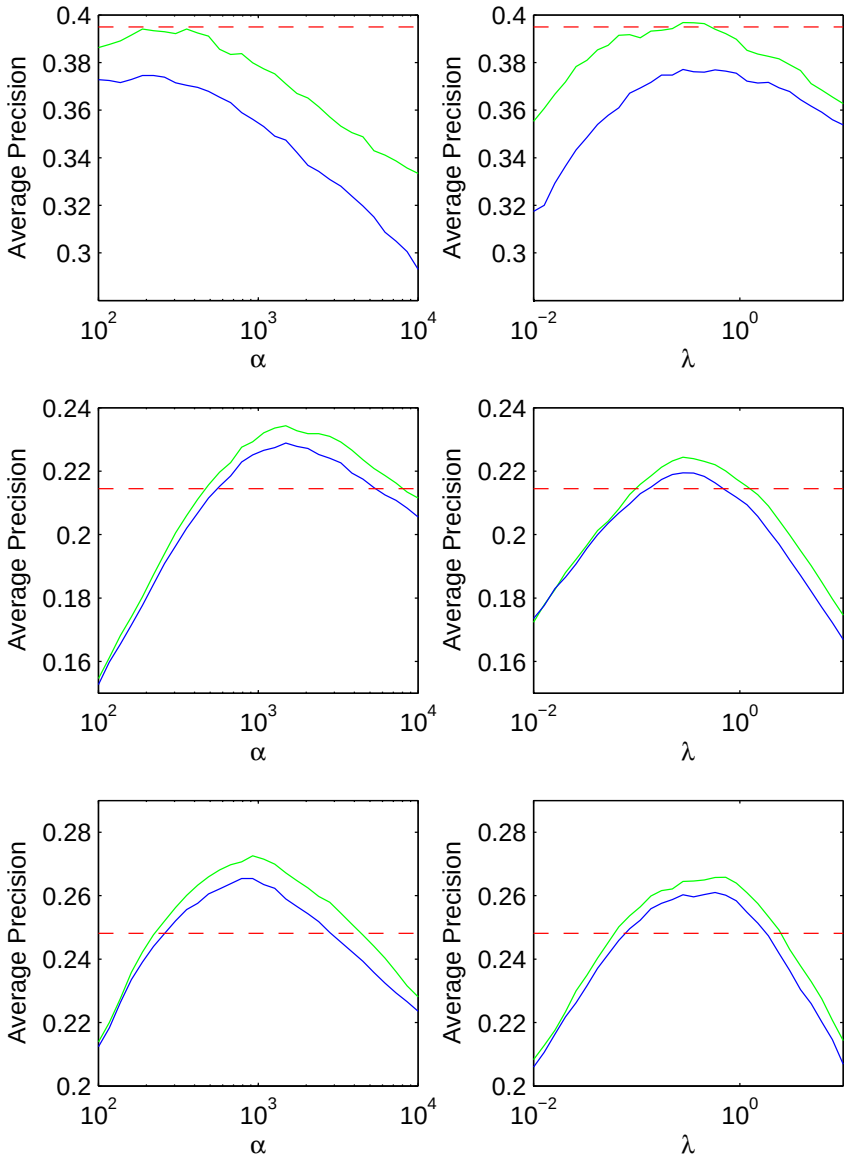


Figure 4.3: Average precision results. Left: the hierarchical Dirichlet model described in Section 4.3.1 (green) and Dirichlet smoothing with the maximum likelihood collection model (blue) as a function of the Dirichlet parameter α. Right: mixture models with the document frequency distribution (green) and the maximum likelihood collection model (blue) as a function of the mixing ratio λ. Results for BM-25 are shown as dashed horizontal lines.

Cranfield:
1400 docs
225 queries

TREC 7:
528,155 docs
50 queries

TREC 8:
528,155 docs
50 queries

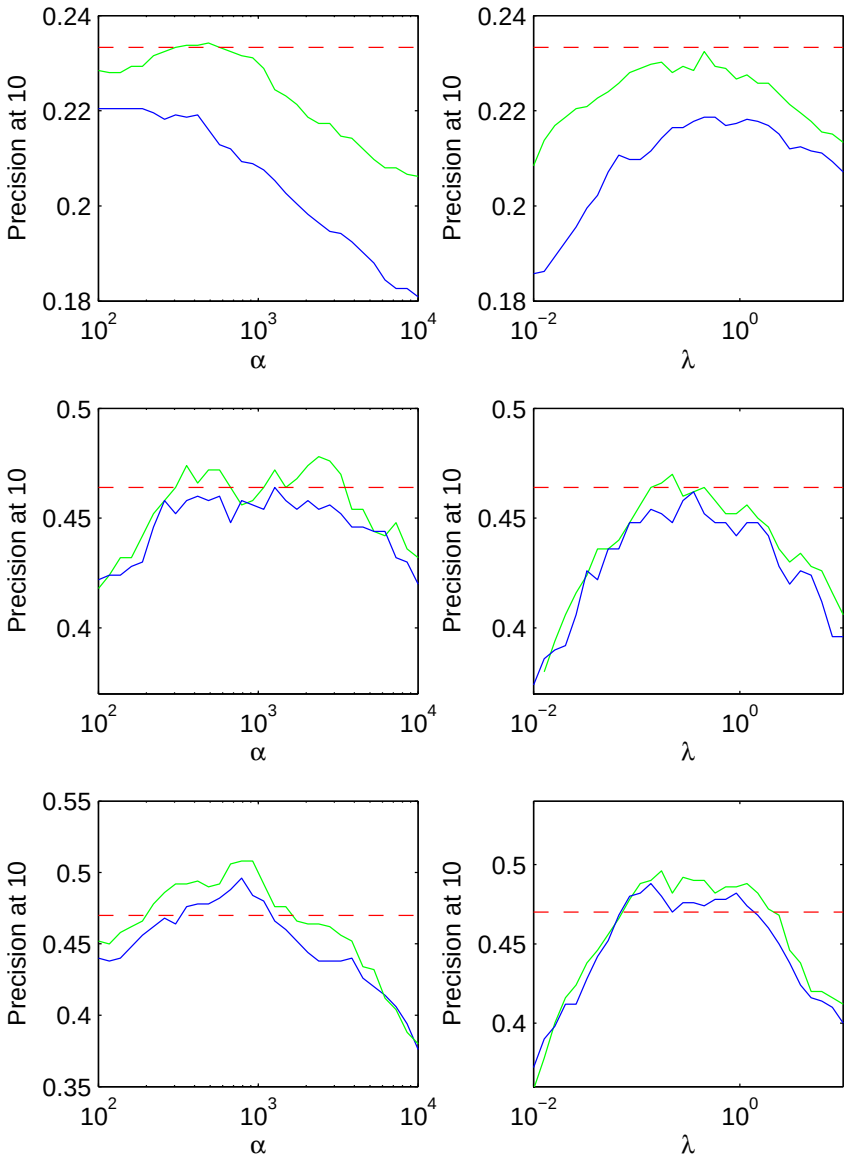


Figure 4.4: Mean precision results at 10 documents. Left: the hierarchical Dirichlet model described in Section 4.3.1 (green) and Dirichlet smoothing with the maximum likelihood collection model (blue) as a function of the Dirichlet parameter α. Right: mixture models with the document frequency distribution (green) and the maximum likelihood collection model (blue) as a function of the mixing ratio λ. Results for BM-25 are shown as dashed horizontal lines.

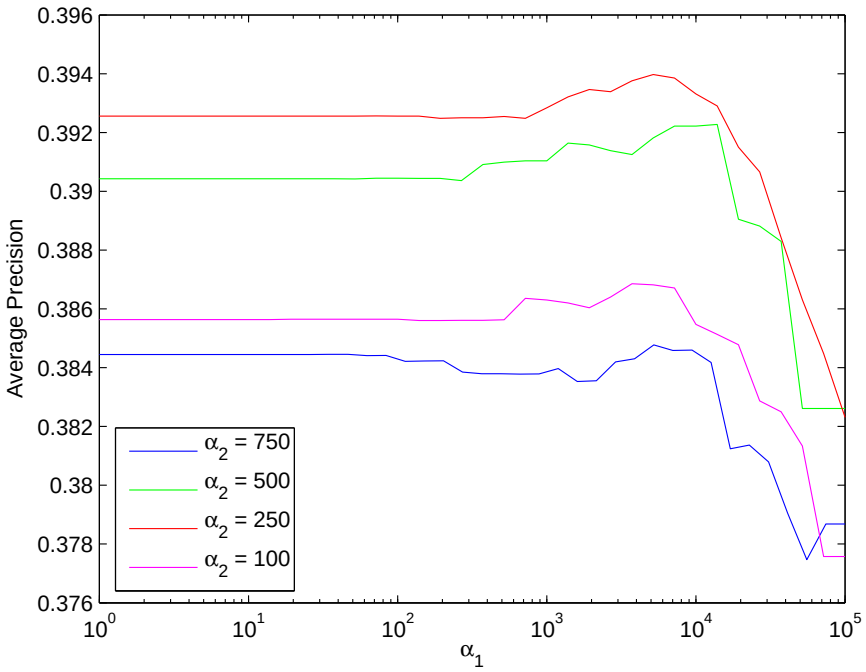


Figure 4.5: Variation of the average precision for the Cranfield data set as a function of α_1. Curves are shown for a variety of values of α_2.

4.4.2 Effect of varying α_1

The experiments described in the previous section kept α_1 fixed whilst varying α_2. As α_1, which describes the probability of escaping from the oracle to the top level prior, is expected to have less of an effect on the predictions, this restriction is reasonable, but it is of course valuable to see what happens if α_1 is varied. To evaluate the effects of this parameter, we calculated the average precision for the Cranfield data set as both α_1 and α_2 were varied. The results are shown in Figure 4.5. Over a wide range of values, between approximately 1 and 10^3, no change is seen due to changes in α_1. Between 10^3 and 10^4 a small improvement in performance is observed, and above this range the performance drops off sharply.

4.4.3 Precision–recall curves

To get a fuller picture of the performance of the new method, precision–recall curves were plotted. Curves for the TREC-7 and -8 query sets are shown in Figures 4.6 and 4.7, using $\alpha_1 = 750$ and $\alpha_2 = 1250$, with BM-25 (using the default parameters as described above) and the Twenty One model (a mixture model using document frequencies as the base distribution and $\lambda = 17/3$) shown for comparison. Although the curves are very close, careful examination shows that BM-25 performs well at low recall, while the Twenty One model gives slightly better performance towards the tail end of the results, where recall is high. The hierarchical

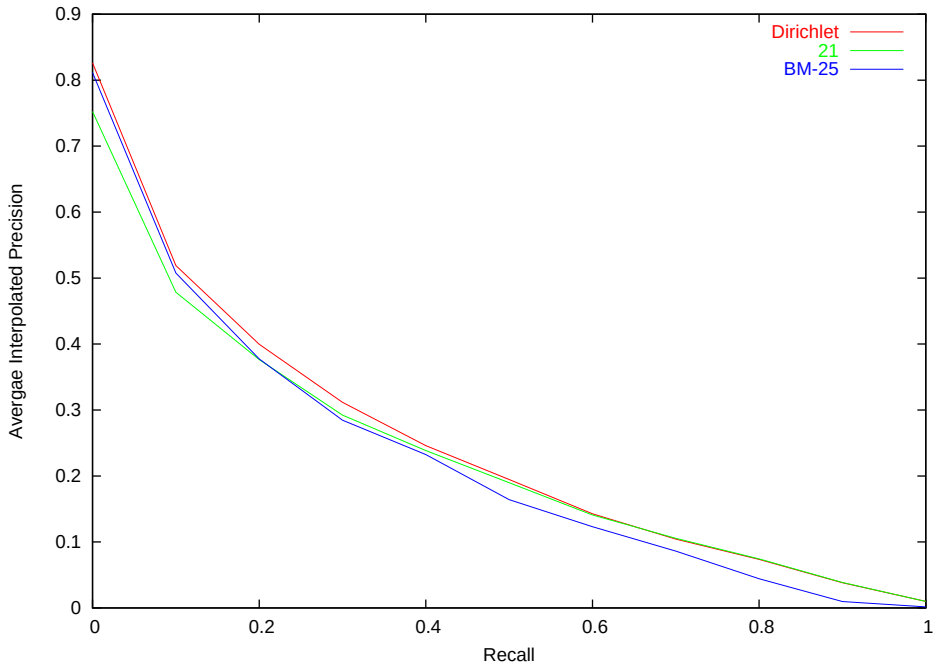


Figure 4.6: Precision–recall curve for the TREC 7 query set.

Sections	Abbreviation	Mean Length	
		TREC7	TREC8
Title	T	2.44	2.44
Title + Description	TD	16.5	16.2
Title + Description + Narrative	TDN	57.0	51.3

Table 4.1: The three query sets used to investigate the effect of query length, with mean number of terms per query.

model however fits the envelopes of the two curves, matching the better performance at both extremes.

4.4.4 Variation of query length

The TREC query sets include three sections for each query: title, description and narrative. By combining these sections it is possible to create queries of varying length and thus investigate the effect on retrieval performance (see Table 4.1). We investigated this effect for the three models described in the previous section. For this experiment the parameters of the hierarchical Dirichlet model were held fixed at $\alpha_2 = 1250$ and $\alpha_1 = 750$. Results for experiments varying the query length are shown in Figure 4.8, with numerical results for the TDN query set given in Table 4.2. As would be expected, the results show that retrieval performance increases with the length of the query. The hierarchical Dirichlet model gives the highest performance in all cases.

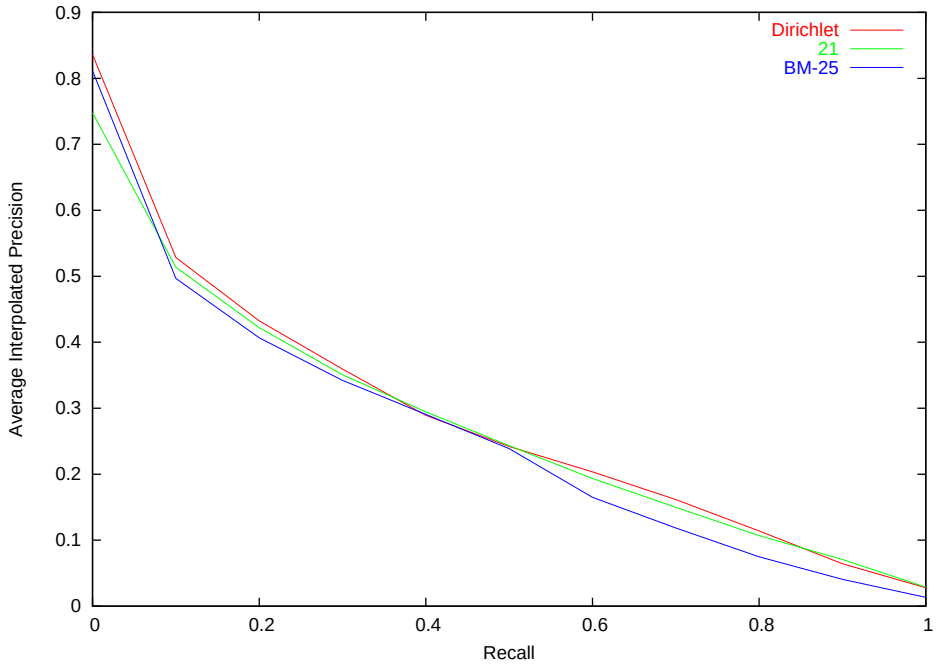


Figure 4.7: Precision–recall curve for the TREC 8 query set.

	TREC-7	**TREC-8**
BM-25	0.21	0.25
Twenty-One	0.22	0.26
Dirichlet	*0.23*	*0.27*

Table 4.2: Average non-interpolated precision scores over top 1000 documents for TREC-7 and TREC-8 *ad hoc* tasks. The Dirichlet results used $\alpha_2 = 1250$ and $\alpha_1 = 750$. BM-25 used $k_1 = 1.2$, $k_3 = 7$ and $b = 0.75$ and the Twenty-One model used $\lambda = 17/3$.

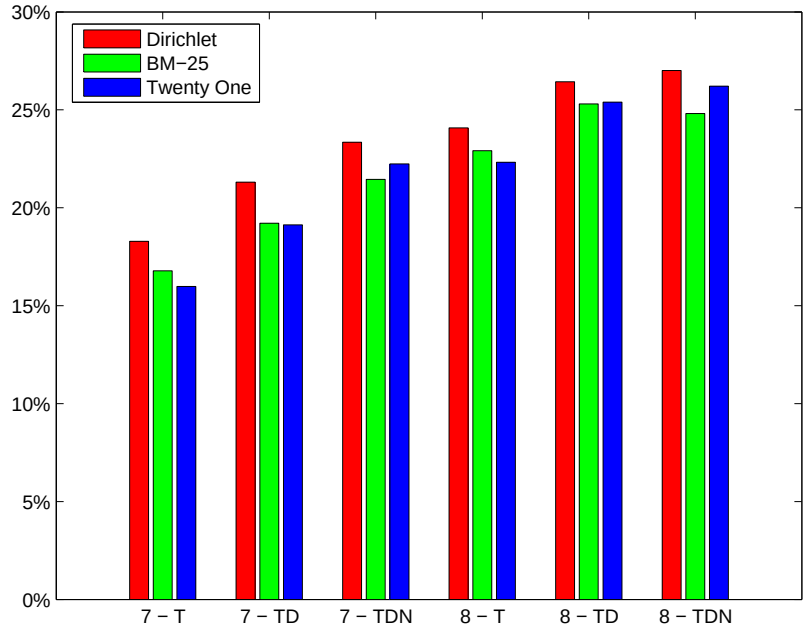


Figure 4.8: Summary of results for the TREC 7 and 8 query sets.

4.5 Deeper hierarchies and passage retrieval

This section extends the retrieval framework developed above to passage retrieval. The goal in passage retrieval is to extract particular passages from those documents which are especially relevant. At the most basic level, this is of use in identifying sections of the document to present to the user when displaying the results of the query. Furthermore, there is evidence that this approach can improve search results [39]. As many documents span multiple topics or contain large amounts of 'boilerplate' text such as menus in hypertext documents, legal disclaimers and so on, it is understandable why passage retrieval is worth pursuing.

The notion of a passage is not well defined, and several approaches have been used in the past. Typically passages are either author-specified, corresponding to sentences, paragraphs, marked sections in the text and so on, or are generated automatically, for example with a fixed length. Passages may or may not overlap. Approaches have also been tried which aim to consider all regions of the text as potential passages. For further background in this area, see [53], [39] and [16], as well as references therein.

Previous evaluations of language model based passage retrieval [53] have treated each passage as a separate document. In this section, the fact that passages belong to documents is made explicit by extending the Dirichlet model to include another level in the hierarchy. In the extended model a distribution over terms, $\boldsymbol{q}_{jk}$, is maintained for passage k in document j. These distributions are generated from a Dirichlet distribution which is shared by all passages

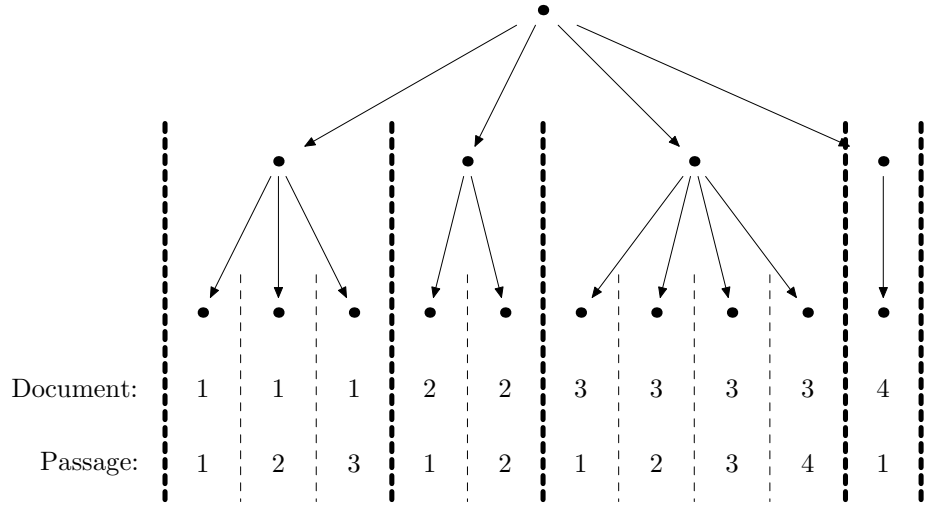


Figure 4.9: Schematic representation of the three-level model used for passage retrieval, showing how distributions in the hierarchy are allocated to passages. Each circle represents a distribution over terms. The distributions for each of the passages in a document share a common parent distribution. These distributions, of which there is one per document, share a single global distribution for the whole collection.

in a given document, and the parameter for that distribution, $\alpha \boldsymbol{q}_j$ is based on a Dirichlet shared by the whole collection as was previously the case.

$$\begin{aligned} \boldsymbol{p} &\sim \text{Dirichlet}\left(\alpha_1 \boldsymbol{u}\right) && (4.27)\\ \boldsymbol{q}_j &\sim \text{Dirichlet}\left(\alpha_2 \boldsymbol{p}\right) && (4.28)\\ \boldsymbol{q}_{jk} &\sim \text{Dirichlet}\left(\alpha_3 \boldsymbol{q}_j\right) && (4.29)\\ t_l &\sim \text{Categorical}\left(\boldsymbol{q}_{d_l p_l}\right) && (4.30) \end{aligned}$$

The model is equivalent to that shown in Figure 2.10 as a Bayesian network, with J indexing documents, K indexing passages within the documents and L indexing terms within each passage. See Figure 4.9 for a more schematic illustration of the model.

Using the minimal oracle approximation and ignoring any dependency of the predictive distribution on previously observed query terms as before, the probability of a query term t is given by

$$\Pr\left(t \mid d, p\right) = \frac{1}{\alpha_3 + N_p}\left(n_t\left(t,p\right) + \alpha_3 \frac{1}{\alpha_2 + N_d}\left(n_p\left(t,d\right) + \alpha_2 \frac{1}{N + \alpha_1}\left(n_d\left(t\right) + \frac{\alpha_1}{|\mathcal{V}|}\right)\right)\right) \quad (4.31)$$

where $n_t\left(t,p\right)$ is now the term frequency *within the passage*, $n_p\left(t,d\right)$ is the *passage frequency*, defined as the number of passages in document d containing the term at least once and $n_d\left(t\right)$

is the document frequency as before. N_p, N_d and N are the sums of these quantities over all terms in the vocabulary. Defining $\hat{p}(t)$ in a similar way to before,

$$\hat{\mathrm{p}}(t) = \frac{n_d(t) + \frac{\alpha_1}{|\mathcal{V}|}}{N + \alpha_1} \tag{4.32}$$

(4.31) can be re-arranged to give

$$\Pr(t \mid \boldsymbol{d}, \boldsymbol{p}) = \frac{1}{\alpha_3 + N_p} \frac{1}{\alpha_2 + N_d} \left(\frac{n_t(t,p)(N_d + \alpha_2) + \alpha_3 n_p(t,d)}{\alpha_2 \alpha_3 \hat{p}(t)} + 1 \right) \alpha_2 \alpha_3 \hat{p}(t) \tag{4.33}$$

Taking logarithms and ignoring terms which do not depend on the document, we obtain

$$\log \Pr(t \mid \boldsymbol{d}, \boldsymbol{p}) = \log \frac{1}{\alpha_3 + N_p} + \log \frac{1}{\alpha_2 + N_d} + \log \left(\frac{n_t(t,p)(N_d + \alpha_2) + \alpha_3 n_p(t,d)}{\alpha_2 \alpha_3 \hat{p}(t)} + 1 \right) \tag{4.34}$$

Finally, this can be summed over all terms in the query to give

$$R(\boldsymbol{p}, \boldsymbol{q}) = N_q \left(\log \frac{1}{\alpha_3 + N_p} + \log \frac{1}{\alpha_2 + N_d} \right) + \sum_i \log \left(\frac{n_t(t,p_i)(N_d + \alpha_2) + \alpha_3 n_p(t,d_i)}{\alpha_2 \alpha_3 \hat{p}(t_i)} + 1 \right) \tag{4.35}$$

As before, the relevance score can be interpreted as the sum of a document length normalisation term and a set of weights for each term in the query. However, to take into account the additional layer, the forms of these terms have changed.

4.5.1 Query term weighting

The most significant change in the form of the query term weights is that they now longer depend not only on the frequency of occurrence within the passage, but also on the number of times which the term occurs in the whole document. Consider a query for information on the Jaguar make of sports car (as opposed to the animal). A typical document matching this query might consist of reviews of a number of different cars from various manufacturers. The fact that the document is about cars rather than animals would most probably be clear from the introduction to the page, and would not necessarily be reiterated in each passage. This contextual information would be missed if the passage retrieval algorithm considered only terms in the passage itself, but is captured here using references to the whole document.

4.5.2 Document length normalisation

The change to this term is simply that it now takes into account the length of the whole document, not just the passage. As the score function now takes into account terms appearing

anywhere in the whole document, it is arguably appropriate that its length should be included too.

4.5.3 Implementation

The quantities required for this score function are not fundamentally different to those required for the document level algorithm. In essence, two indices are required, one storing counts at the document level and one at the passage level. Under the new score function documents which do not contain a given query term are still assigned zero weight, so an inverted index can be used to rapidly score a collection. Passages within a document containing a given term will still be assigned a non-zero weight even if that term does not appear in the passage itself. We must therefore use the document-level inverted index to find all documents containing each term in the query, then to calculate the score for each passage in that document. As the number of passages in a document is usually relatively small it is still possible to score the whole collection in a reasonable time.

4.5.4 Whole document retrieval

One way of measuring the performance of a passage retrieval method is to perform full document retrieval. While this does not represent the whole story in terms of the abilities of these methods, it does provide a qualitative way to compare different methods. There are a number of ways in which a passage retrieval method can be used to score full documents. We evaluated the following two methods:

- **Most Probable Passage:** Retrieval scores are calculated for all passages in a document, the largest of which is used as the score for the whole document. This approach is similar to that used in [16].
- **Overall Probability:** As the retrieval scores have an interpretation as log probability that the selected passage is relevant (up to a constant offset), and the method fundamentally assumes that precisely one passage is relevant, it is possible to define a score

$$R(d,q) = \log\left(\sum_{p\in d}\exp\left(R\left(p,q\right)\right)\right) \tag{4.36}$$

  which evaluates the probability that any one of the passages in a given document is relevant.

Results for these two approaches on the Cranfield corpus, with sentences being used for the passages, are shown in Figures 4.10 and 4.11. The performance of these two methods is comparable (the 'most probable passage approach' gave 23.6% precision at ten document, compared with 23.3% for the 'overall probability' approach). However, the 'overall probability' approach shows a sharper drop-off as α_3 and α_2 are increased.

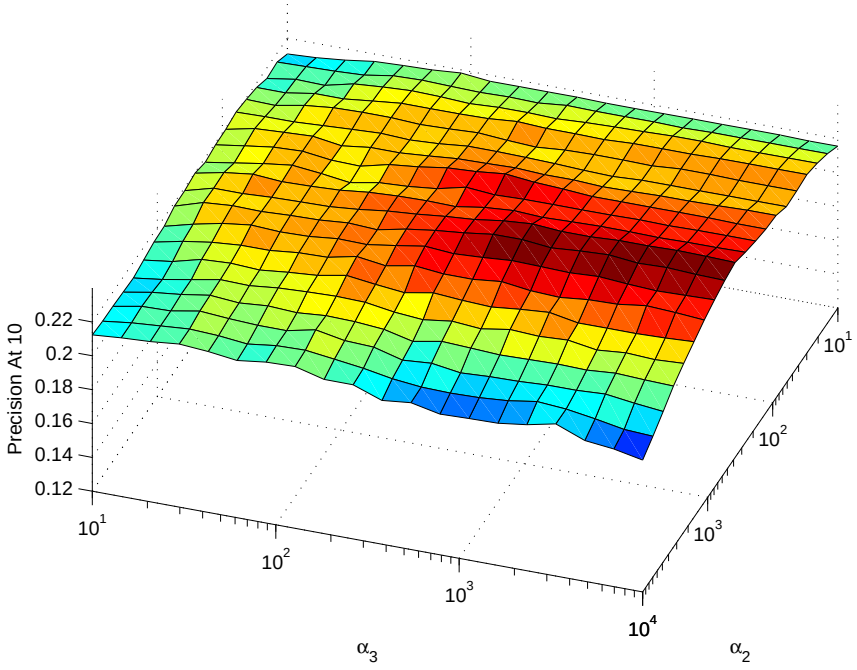


Figure 4.10: Passage retrieval scores for the 'most probable passage' approach.

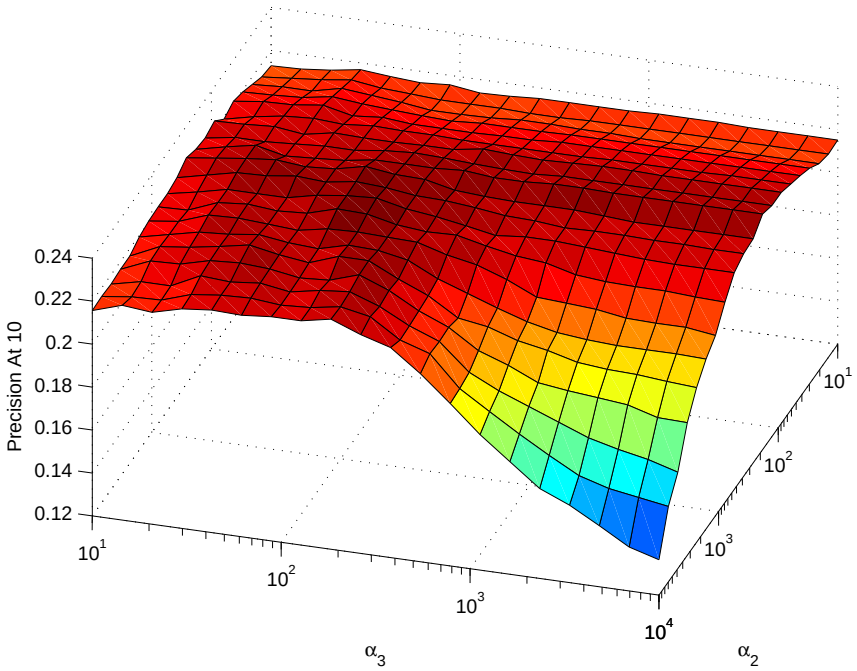


Figure 4.11: Passage retrieval scores for the 'overall probability' approach.

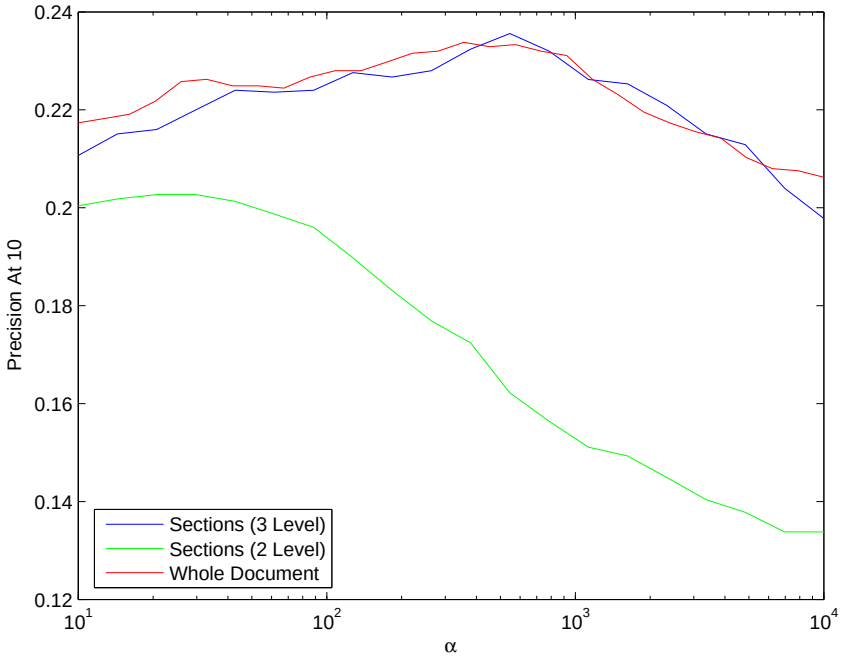


Figure 4.12: Comparison of passage retrieval scores using various methods

Figure 4.12 shows a comparison of three techniques: document retrieval using the method described above with 'most probable passage' scoring, a similar approach using the 2 level model (i.e. treating each passage independently, as if they were separate documents), and whole document retrieval using the approach described in Section 4.3. The results show that the three level approach clearly outperforms the two level approach at the passage level. The whole document approach does give a higher performance overall, but it should be noted that this may depend on the choice of passage types, and that previous studies have found that better performance can be achieved using overlapping passages [16].

4.5.5 Passage selection

A qualitative insight into the performance of the retrieval method can be obtained by looking at the passages which are assigned the highest scores within each of the returned documents. This information may be used to provide a summary of the documents returned, aiding the user in deciding which documents to pursue further. Figures 4.13 and 4.14 show the top ten documents returned for two selected queries from the Cranfield data set, together with summaries generated from the most highly ranked passages within those documents.

In many cases the summaries returned by this method are more informative than the titles of the documents themselves. For example, documents 484 and 578, returned in response to query 3 as shown in Figure 4.13. Neither mention 'composite slabs' in their titles, but do in the returned passages. The ability to return this sort of information is likely to aid the user

Query 3: *What problems of heat conduction in composite slabs have been solved so far.*

Document: 4 (*)
Title: one-dimensional transient heat conduction into a double-layer slab subjected to a linear heat input for a small time internal
Summary: analytic solutions are presented for the transient heat conduction in composite slabs exposed at one surface to a triangular heat rate

Document: 143 (*)
Title: heat flow in composite slabs
Summary:this paper presents the solution of the heat flow problem in composite walls under heat transfer conditions which are typical of uncooled rocket engine walls

Document: 484 (*)
Title: similarity laws for aerothermoelastic testing
Summary: the temperature is determined as a function of position and time in the case of linear heat conduction in a composite slab of ture throughout and the two external surface temperatures are considered to be prescribed functions

Document: 398 (*)
Title: conduction of heat in composite slabs
Summary: conduction of heat in composite slabs

Document: 90 (*)
Title: periodic temperature distribution in a two-layer composite slab
Summary: periodic temperature distribution in a two-layer composite slab

Document: 578
Title: new thermo-mechanical reciprocity relations with application to thermal stress analysis
Summary: this is illustrated by application to one dimensional problems of heating of a homogeneous or composite slab and directly verified by classical methods in the appendix

Document: 89 (*)
Title: periodic temperature distributions in a two layer composite slab
Summary: periodic temperature distributions in a two layer composite slab

Document: 541
Title: the stacking of compressor stage characteristics to give an overall compressor performance map
Summary: radiation cooling due to fourth power radiation from semi infinite solids and finite slabs together with radiation according to newton s law of cooling is then treated

Document: 180 (*)
Title: some problems on heat conduction in stratiform bodies
Summary: some problems on heat conduction in stratiform bodies

Document: 5 (*)
Title: one-dimensional transient heat flow in a multilayer slab .
Summary: one-dimensional transient heat flow in a multilayer slab .

Figure 4.13: Titles and summaries of returned documents for query 3 in the Cranfield data set. Documents marked (*) were marked as relevant in the data set.

Query 156: *what qualitative and quantitative material is available on ablation materials research*

Document: 1095 (*)
Title: an experimental investigation of ablating material at low and high enthalpy potentials .
Summary: qualitative measurements of the effective heats of ablation of several materials in supersonic air jets at stagnation temperature up to 11 000 f...

Document: 1096 (*)
Title: a theoretical study of stagnation point ablation .
Summary: comparisons of the results for several materials tested at the higher heating rates showed graphite to have the lowest ablation rate of all materials tested

Document: 552 (*)
Title: ablation of glassy materials around blunt bodies of revolution .
Summary: ablation of glassy materials around blunt bodies of revolution .

Document: 1064 (*)
Title: plastic stability theory of geometrically orthotropic plates and cylindrical shells .
Summary: the analytical predictions upon inclusion of the pertinent material property values should be applicable to other materials as well as teflon

Document: 1097 (*)
Title: an analytical investigation of ablation .
Summary: an experimental investigation of ablating material at low and high enthalpy potentials

Document: 1098 (*)
Title: a sensor for obtaining ablation rates .
Summary: the most significant result of the analysis is that the effective heat capacity of the ablation material increases linearly with stream enthalpy

Document: 1099 (*)
Title: a five-stage solid fuel sounding rocket system .
Summary: the predicted equilibrium surface temperatures on nonablating surfaces behind an ablating material were in agreement with the values derived from tests conducted with inconel cylinders having teflon hemispherical nose pieces

Document: 81 (*)
Title: theoretical investigation of the ablation of a glass-type heat protection shield of varied material properties at the stagnation point of a re-entering irbm .
Summary: theoretical investigation of the ablation of a glass type heat protection shield of varied material properties at the stagnation point of a re entering irbm

Document: 1278
Title: turbulent heat transfer on blunt-nosed bodies in two-dimensional and general three-dimensional hypersonic flow .
Summary: in this study of hypersonic ablation the pertinent conservation equations are derived and the simultaneous processes of diffusion convection...

Document: 1116
Title: stability of orthotropic cylindrical shells under combined loading .
Summary: the increasing use of fiber and whisker reinforced materials makes necessary the availability of methods of analyzing cylinders and cones composed of an orthotropic material

Figure 4.14: Titles and summaries of returned documents for query 156 in the Cranfield data set. Documents marked (*) were marked as relevant in the data set.

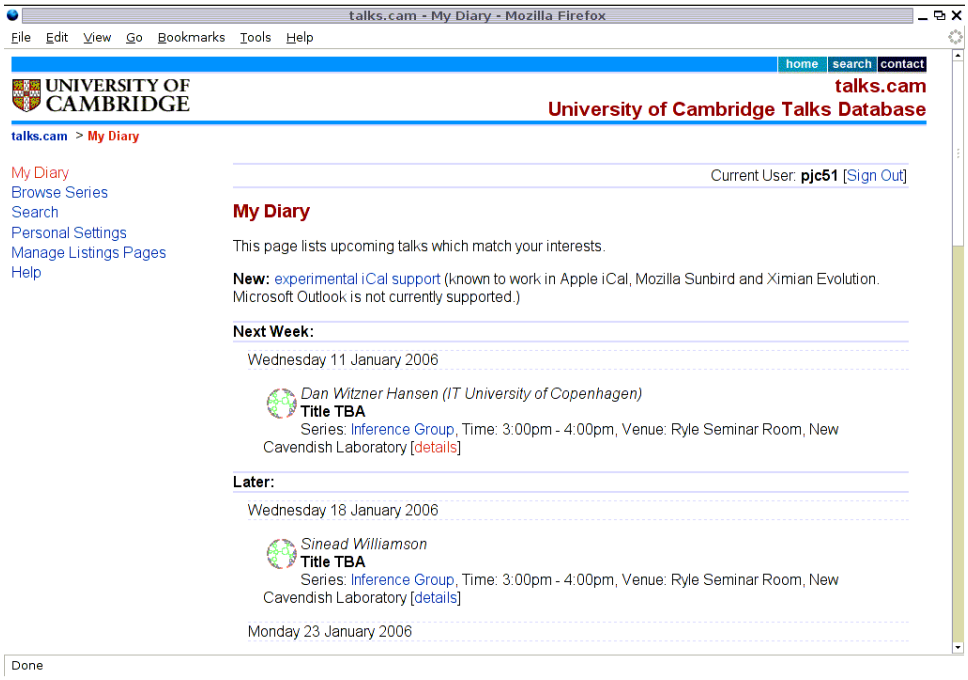


Figure 4.15: Screen capture of the 'My Diary' section of *talks.cam*. This page shows a personalised listing of future events, based on series to which the user has subscribed. Alternatively, the user may browse a directory of all events, or search using the facility described below.

in determining the relevance of the document without having to read it in full.

4.6 Application to a seminar database

Large academic institutions such as the University of Cambridge typically host hundreds of seminars and similar events each year. Currently, these are usually advertised through a combination of departmental web sites, postings on notice boards and email lists. The lack of a unified mechanism for advertising events results in an information overload for many people and sets up a barrier to inter-disciplinary interaction.

To address these problems, we created the *talks.cam* system [111], a web-based database of talks and seminars taking place in the University of Cambridge. The aim of the service is to provide a central location for listing such events, allowing users of the service to construct their own diaries containing events which are of interest to them (see Figure 4.15). In approximately 1 year, details of 350 talks were entered covering 35 seminar series.

In order to allow users to find events, a search facility was implemented using the information retrieval method described in Section 4.3. Each document in the index represents a single event, and consists of the title and abstract of the seminar as well as the name of the speaker. Porter stemming and stop-word removal were used as in the examples described above. The search facility allows searches to be restricted to talks occurring in the future or in the past.

4.7 Conclusions

This chapter has developed a text retrieval framework based on the hierarchical Dirichlet model introduced in Chapter 2. The model is different in a number of ways to existing techniques. Most importantly, the approach provides justification for the use of inverse document frequency statistics, which until now have been either absent or introduced as a heuristic in the language modelling approach to information retrieval. The suggested method was shown to perform well, giving better retrieval performance than existing methods on a number of test tasks.

The hierarchical model was further extended to the task of passage retrieval. This results in a score function for passages which not only considers the appearance of terms from the query in the passage itself, but also their appearance in the wider context of the whole document. The new passage retrieval method shows promising performance when compared to other approaches to this problem.

Chapter 5

Dasher As A Search Tool

5.1 Introduction

The previous chapter described a probabilistic approach to information retrieval in large collections. While it is clear from the results presented above that this approach is capable of providing very high performance in terms of the search results themselves, it is possible to take the idea further and use probability theory to make improvements to the way in which the search string is entered and the results are presented to the user. In Section 1.3.1, the Dasher user interface was described as a method of text entry making use of a probabilistic language model. In fact, it is possible to use Dasher in any context where navigation of data organised as a tree is necessary, and where estimates are available of the probability that the user will want to select a particular item.

This chapter investigates the application of the Dasher user interface to search tasks. For purposes of simplicity, the task is limited to that of 'exact search' — in other words, tasks where the user wishes to obtain information exactly matching a query. Two such tasks are considered — firstly searching for substrings within a text document, and secondly searching for matching records in a database. The analysis presented here concentrates on theoretical aspects of these applications.

5.2 Interactive search

Given a text document, it is often desirable to be able to locate all occurrences of a particular substring. Whilst many text processing applications provide this feature, in the majority of cases the search is non-interactive. In other words, the user provides the full substring, the search is performed, and the user then selects one of the possibly multiple matches. A few applications however provide an interactive search function. In this case, the search is performed after each symbol in the substring has been entered. Possible matches are indicated in the document, and the user may choose either to select a match from the current list, enter more symbols in order to refine the search, or delete symbols in order to make it less specific. Examples of applications providing this functionality include the GNU Emacs text editor

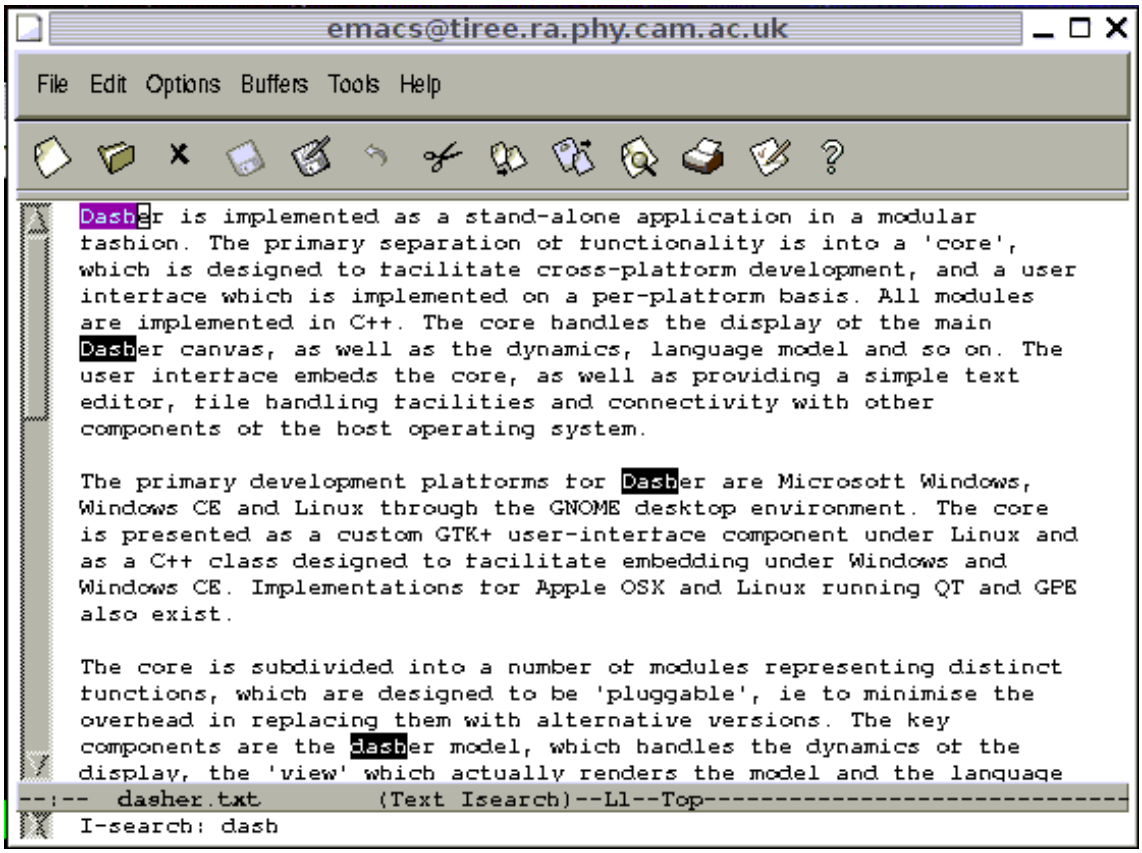


Figure 5.1: The interactive search (I-search) feature in the GNU Emacs text editor. Having entered the substring 'dash' all occurrences of that phrase in the document are highlighted.

[106] (see Figure 5.1) and the Mozilla Firefox web browser [109].

While such an interactive search feature is clearly preferable over the traditional non-interactive version, the use of free keyboard entry is limited in the sense that it offers no suggestion in advance as to which substrings occur in the document; when entering a new character to refine the search, the user is informed of the success of the search only after the character has been entered. This limitation can be addressed by using the Dasher interface.

In order to implement a search tool using Dasher, the problem must be recast into that of selecting a node from a tree structure whose branches are weighted by probabilities. This representation follows naturally from the ordering of all possible substrings in dictionary order. In other words, using Dasher as an interface for search can be considered equivalent to using it for text entry, except that the language model assigns zero probability to all strings which are not present in the document. It is initially assumed that the distribution over possible strings is uniform, so the probability associated with each node is simply proportional to the number of substrings for which that node is a prefix.

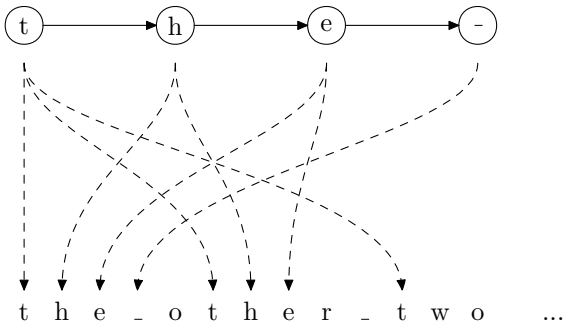


Figure 5.2: An illustration of the data structure used in search Dasher. The stored text, 'the_other_ two...' is shown along the bottom of the diagram, while the chain at the top represents one path from root to node through the trie. Each node stores a list of locations in the text corresponding to the end points of matches. From left to right these are 't', 'th', 'the' and 'the_' respectively. The leaf node, corresponding to '_', has only one item in its list meaning that the string is deterministic from this point on. Rather than increasing the depth of the trie, to find the subsequent symbols it is only necessary to follow the stored text from the position indicated by the unique pointer.

5.2.1 Data structure

The structure used to represent the document in a way which facilitates construction of the Dasher interface is essentially the same as that used in the PPM* text compression algorithm, a member of the PPM family described in Section 2.3.5 which makes use of unbounded context lengths [20]. Data is stored in a trie very similar to that used in Chapter 2, but where each node contains a list of references to locations in the text, which is also stored linearly in memory. The references point to locations in the text corresponding to the last symbol in instances of the N-gram represented by the location in the trie. There is no maximum depth for the trie, but it is terminated whenever a node is reached which contains only one pointer to the text. After this point, there is only one possible continuation of the substring, which can be found simply by following the text itself from the target of the last pointer. The data structure is illustrated in Figure 5.2. By maintaining a list of 'active' nodes in the trie, i.e. those which contain pointers to substrings terminating at the last character read, the structure can be generated in a single parse of the file.

Dasher itself maintains a trie structure corresponding to the hierarchy of boxes displayed on the screen. Each node in the Dasher trie represents one box, and the children of each node are the boxes contained within it. Thus there is a one to one correspondance between nodes in the data structure representing the text being searched, and those in the data structure used by Dasher. Each node in the Dasher trie needs to maintain a pointer to either a node in the data trie, or a position in the text. For Dasher nodes pointing to the data trie, the process of populating their children as the user zooms in is simply a matter of iterating over the children in the data trie. Each child Dasher node gets a pointer to the corresponding data node, or if that node only points to one location in the text, the Dasher node is given

Field	Description	Size (bytes)
Symbol	The symbol in the character set represented by the node.	1
Sibling	A pointer to the next child of the node's parent, or NULL to indicate that this is the last child.	4
Child	A pointer to the first child of the node, or NULL to indicate that no children exist.	4
Pointer	A pointer to a linked list of pointers to locations in the text.	4
	Total:	13

Table 5.1: Sizes of the fields required for storage in the data structure used in Search-Dasher, assuming a 4 byte pointer size.

a pointer to that location. Dasher nodes which already point to the text only have a single child, which has a pointer to the next location in the text[1].

We constructed a prototype implementation of this system, referred to as *Search-Dasher*. Screen captures of this prototype is shown in Figure 5.3. The figures show the sequence of events as the search term 'the' is entered one character at a time. After each character has been entered, all matches in the document are highlighted in the pane to the left of the screen.

5.2.2 Storage requirements

An important consideration when implementing a search facility is the size of storage required to maintain the data in a way which facilitates the search process. The method described above has two components. Firstly, each node in the trie contains a data structure needed to store the details of the trie itself. An illustration of the size of this structure is given in Table 5.1. Associated with each node is a variable length array of pointers into the text, which in order to facilitate construction is stored as a linked list. Each pointer will therefore use 8 bytes, with 4 for the pointer into the text itself and 4 for the link pointer. The total storage requirements are therefore

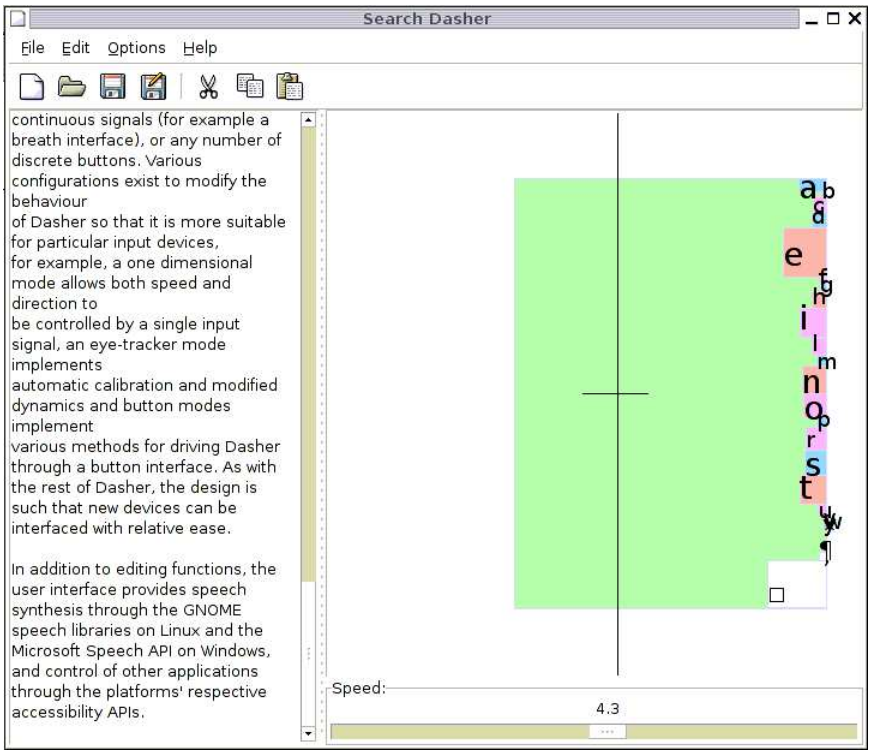
$$b_t = 13 \cdot n_n + 8 \cdot n_p \tag{5.1}$$

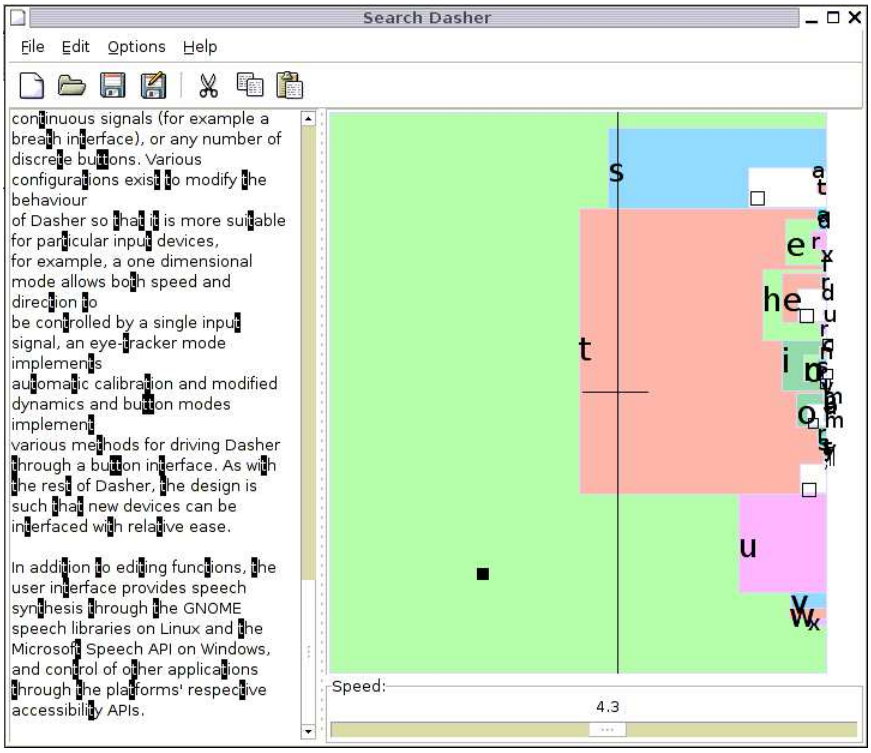
where n_n is the number of nodes and n_p is the number of pointers. Storage requirements are shown using experimentally obtained values for n_n and n_p in Figure 5.4.

The overall storage requirements are fairly large. Documents of 10kB in length will require of the order of 65 times their length in additional storage for the index. This does not necessarily limit the usefulness of this method however - a 10,000 byte document will require 640,000 bytes of storage of storage for the index, which is still quite small by the standards of contemporary desktop PCs. A document of length 100,000 will require 10^7 bytes of storage.

In some cases it is necessary to be able to search for only entire words. In this case,

[1] In practice additional nodes will be added to correspond to 'escaping' from search mode and performing some action on the strings which have been selected.


Search Dasher
File Edit Options Help
continuous signals (for example a breath interface), or any number of discrete buttons. Various configurations exist to modify the behaviour
of Dasher so that it is more suitable for particular input devices,
for example, a one dimensional mode allows both speed and direction to
be controlled by a single input signal, an eye-tracker mode implements
automatic calibration and modified dynamics and button modes implement
various methods for driving Dasher through a button interface. As with the rest of Dasher, the design is such that new devices can be interfaced with relative ease.
In addition to editing functions, the user interface provides speech synthesis through the GNOME speech libraries on Linux and the Microsoft Speech API on Windows, and control of other applications through the platforms' respective accessibility APIs.
Speed:
4.3


Search Dasher
File Edit Options Help
continuous signals (for example a breath interface), or any number of discrete buttons. Various configurations exist to modify the behaviour
of Dasher so that it is more suitable for particular input devices,
for example, a one dimensional mode allows both speed and direction to
be controlled by a single input signal, an eye-tracker mode implements
automatic calibration and modified dynamics and button modes implement
various methods for driving Dasher through a button interface. As with the rest of Dasher, the design is such that new devices can be interfaced with relative ease.
In addition to editing functions, the user interface provides speech synthesis through the GNOME speech libraries on Linux and the Microsoft Speech API on Windows, and control of other applications through the platforms' respective accessibility APIs.
Speed:
4.3

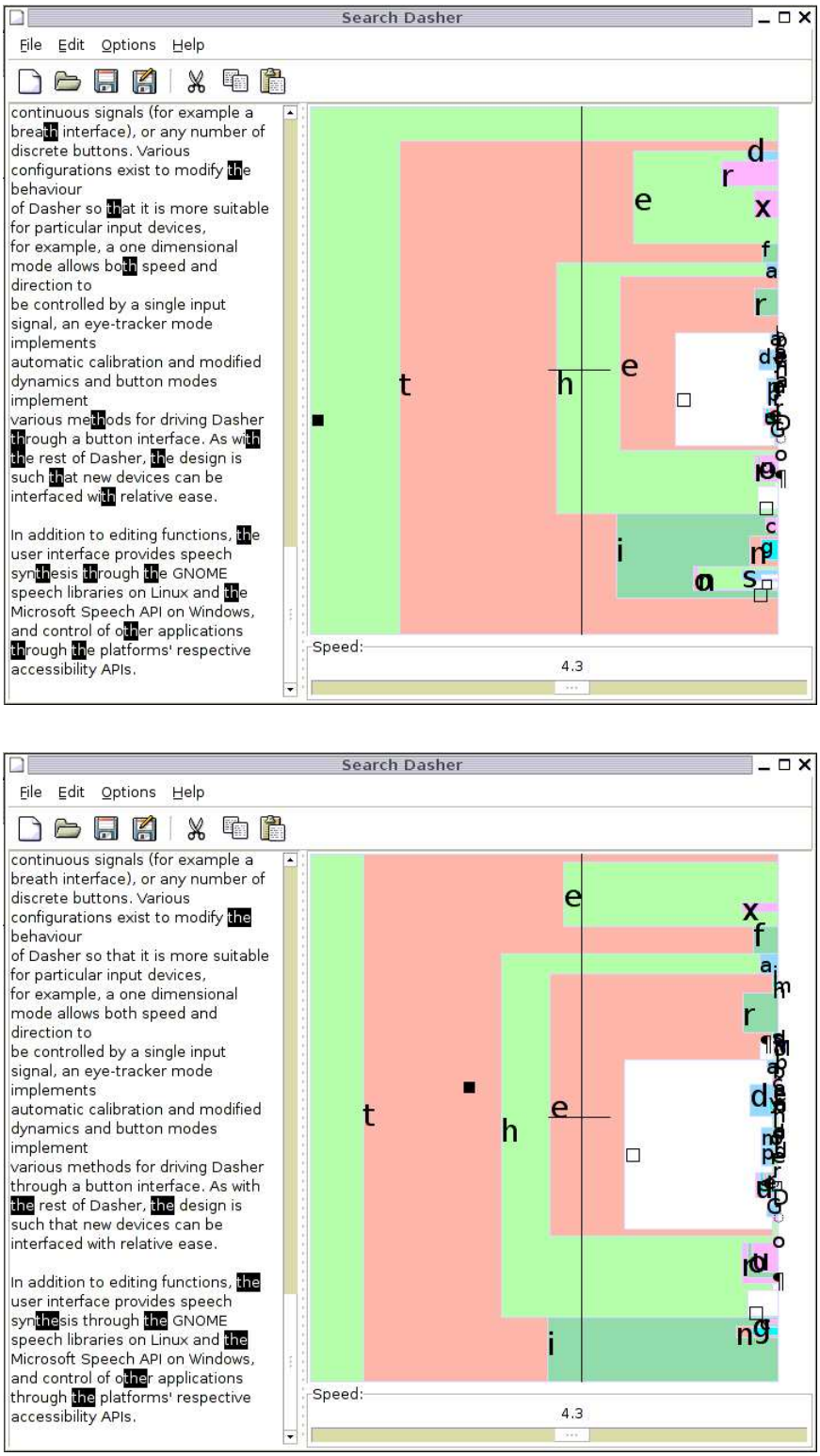


Figure 5.3: Screen captures showing the prototype Search-Dasher implementation

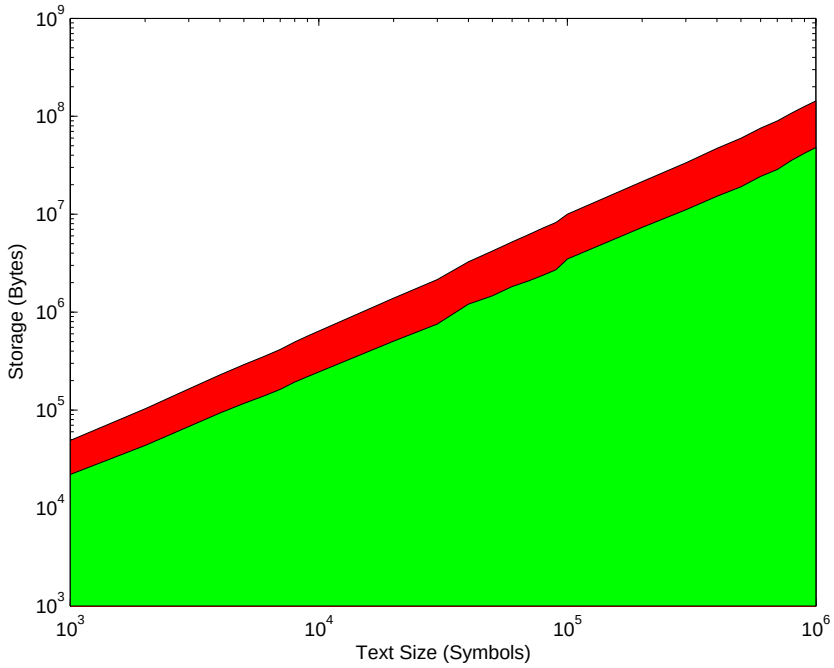


Figure 5.4: Storage requirements for the search index as a function of document length. The green area indicates the storage for the per-node data structure and the red area indicates the storage for the variable length pointer arrays.

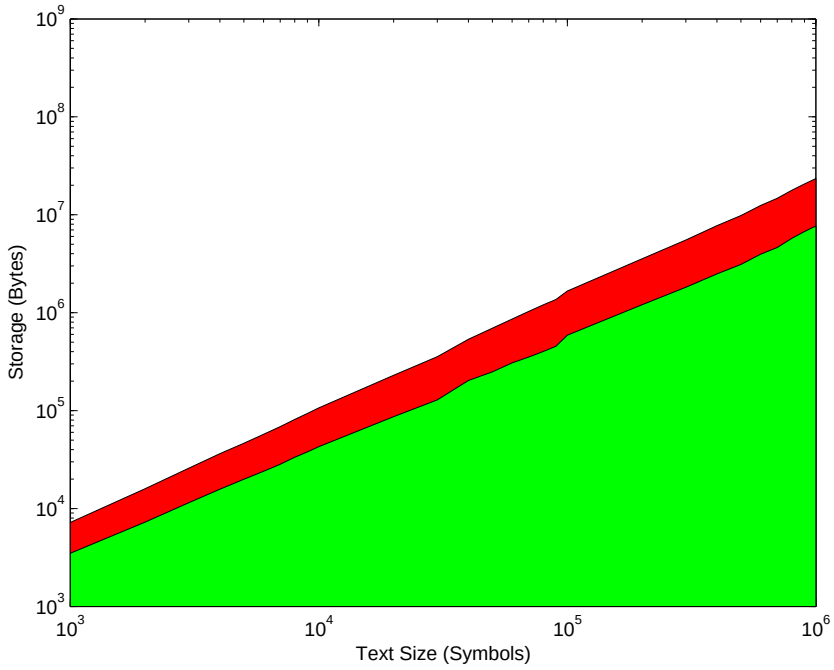


Figure 5.5: Storage requirements for the search index as a function of document length if only word prefixes are stored. The green area indicates the storage for the per-node data structure and the red area indicates the storage for the variable length pointer arrays.

substrings need be stored only if they are the prefix to words in the document. Performing the same calculation on this reduced set of strings yields significantly smaller requirements: a 10,000 byte document will produce an index of the order of 110,000 bytes and a 100,000 byte document of the order of 1.7×10^6 bytes. Storage requirements over a range of file sizes are shown in Figure 5.5.

5.2.3 Dynamic node allocation

It is possible to improve on the storage requirements by making a trade-off between pre-computation and computation at the time of Dasher node creation. To achieve this, rather than truncating the trie when the substring becomes unique, it is truncated at the point where the substring occurs no more than some small fixed number of times, $n_{\max}$. Each node in the Dasher trie must now be able to store either a pointer into the data trie, or a list of at most $n_{\max}$ pointers, each of which must be followed in the text when that node's children are created. Fortunately only Dasher nodes which are required to render the display are instantiated, so it is expected that the additional run time overhead will be low.

As an illustration, consider the case for $n_{\max} = 2$ on the data shown in Figure 5.2. In this case, the trie would be truncated below the node corresponding to h, which has 2 pointers into the data. To populate the children of the corresponding Dasher node, the text itself

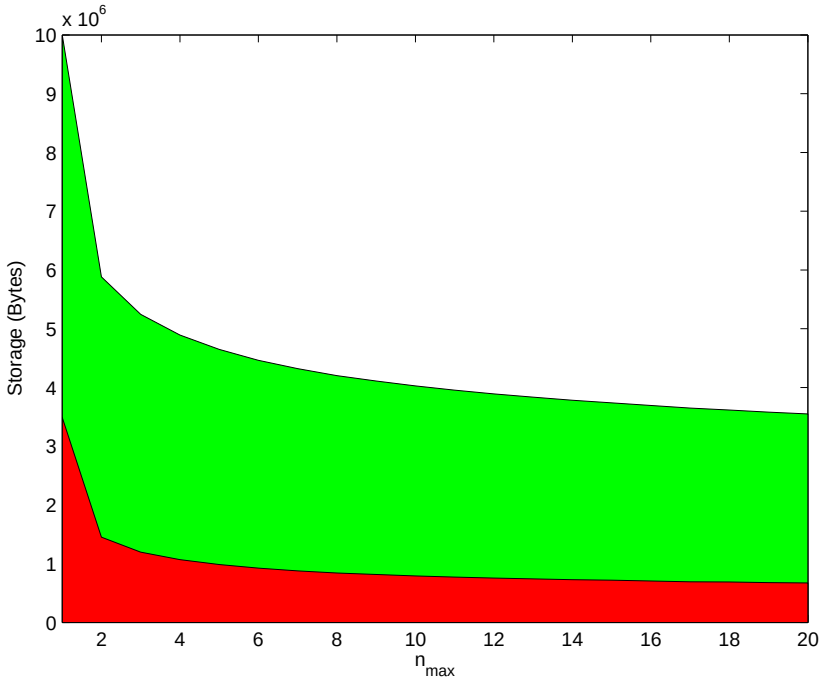


Figure 5.6: Storage requirements using dynamic node allocation as a function of $n_{\max}$ for a 100,000 byte text with full substring indexing. Storage for the trie itself is shown in green and that for the pointer lists is shown in red.

would be examined to find the possible continuations (both 'e' in this case). These pointers must both be maintained in order to populate the next level children, which then differ ('␣' and 'r').

Results for the possible storage savings based on experimental data are given in Figure 5.6 and numerically in Table 5.2. By setting $n_{\max} = 4$ it is possible to save more than 50% of the storage space, suggesting that this would give good performance in practice. Increasing $n_{\max}$ beyond this level results in diminishing returns, so the added computational cost will be less worthwhile.

5.2.4 Distributions over strings

The previous section assumed a uniform distribution over strings. It is of course, not necessary to make this assumption, and in many cases may be desirable to avoid it. In the simplest case, if the full text is indexed, it may be desirable to increase the probability of substrings which are word prefixes. In more structured documents, paragraph prefixes, headings and so on may also be more likely search candidates and can have their probability increased as appropriate. An example towards the more structured end of the scale is a computer source code editor, where it may be desirable to increase the probability of, for example, locations in the text corresponding to the definition of subroutines.

$n_{\max}$	**Storage Reduction**
1	0%
2	41%
3	48%
4	51%
5	54%
6	55%
7	57%
8	58%
9	59%
10	60%

Table 5.2: Percentage savings in storage requirements as a function of $n_{\max}$ for a 100,000 byte text with full substring indexing.

If each string is given a weight, corresponding to the un-normalised probability, this weight can be stored alongside each of the pointers. When populating children, the relative sizes of the boxes can be made proportional to the sums of the weights for all pointers in the corresponding child nodes. This will increase the storage size slightly, but depending on the resolution required it may only be necessary to add a few bits per entry.

5.3 Contact list lookup

One significant advantage of Dasher over many other text entry systems is that it requires only a pointing device or a small number of buttons, rather than a large keyboard. This property is particularly appropriate for text entry on small devices such as mobile telephones and personal digital assistants (PDAs). This section considers the task of selection from a database where each record is indexed by a short string. An example of an application involving this task might be the selection of a record from the contacts database stored on a telephone, where records are indexed by name.

A popular method of text entry on such devices is "predictive text entry" such as the T9 system [112]. In these systems, a small number of keys are mapped to the full alphabet, as well as numbers and some punctuation symbols. As the mapping is one-to-many, it is often necessary to disambiguate between multiple strings corresponding to the same key sequence. For free text entry this task is performed using a dictionary, possibly with some information about the relative frequencies of different words. Usually two additional keys are required for this task - one to cycle through the list of possible disambiguations and the other to make a selection. Disambiguation is often done on the level of words, so the select key can also carry an implied space. The disambiguation process is illustrated in Figure 5.7. In the selection task, rather than using a free dictionary, the contacts list itself can be used to perform disambiguation.

To analyse this method, we collected a list of the names of 460 members of the University

Key	Symbols
1	Punctuation, 1
2	A, B, C, 2
3	D, E, F, 3
4	G, H, I, 4
5	J, K, L, 5
6	M, N, O, 6
7	P, Q, R, S, 7
8	T, U, V, 8
9	W, X, Y, Z, 9
0	0

Figure 5.7: Left: Predictive text entry on the Sony Ericsson model k500i mobile telephone. The key sequence '739' has been entered, producing a list of possible disambiguations, as well as predictions for the complete word, in this case 'Text'. Right: Mappings from the numeric keypad to sets of symbols on this telephone. Mappings may vary between different models.

of Cambridge Physics Department. For a number of subsets of the list, representing various overall sizes, we calculated the minimum number of key presses required to unambiguously specify each name using T9. Names were stored as the first name followed by the second name, and alternative disambiguations were provided in dictionary order. Accented characters were replaced with their unaccented equivalents, and hyphens were replaced by spaces. In each case we assumed that one additional key press would be required to confirm the selection. We converted the results into times for entry assuming either 1 or 2 key presses per second (this range being obtained through informal observation of typical mobile telephone usage). For comparison, we also calculated the input time using Dasher assuming a uniform distribution over all contacts and a constant information entry rate of 4 bits per second (this figure represents the top end of the range of speeds obtained by users after one hour of experience with Dasher as reported in [96]) . A comparison of the results is shown graphically in Figure 5.8, which indicates that under these assumptions Dasher has the potential to allow significantly greater selection speeds.

An alternative viewpoint is to consider the potential information which could be entered using the same number of keypresses. Assuming a uniform distribution over presses of ten keys this is simply $\log_2(10)$ bits per key press. Multiplying by the mean number of keypresses gives the information which could potentially have been entered. This can be compared to the actual information expressed assuming a uniform distribution over contacts was also calculated. Results of this comparison are shown in Figure 5.9.

5.3.1 Model adaption

Traditional keyboard-based ambiguous text entry systems are essentially bound to a fixed encoding scheme. Although there is scope for some flexibility by varying the order in which possible disambiguations are presented, the mapping from keys to symbols remains fixed.

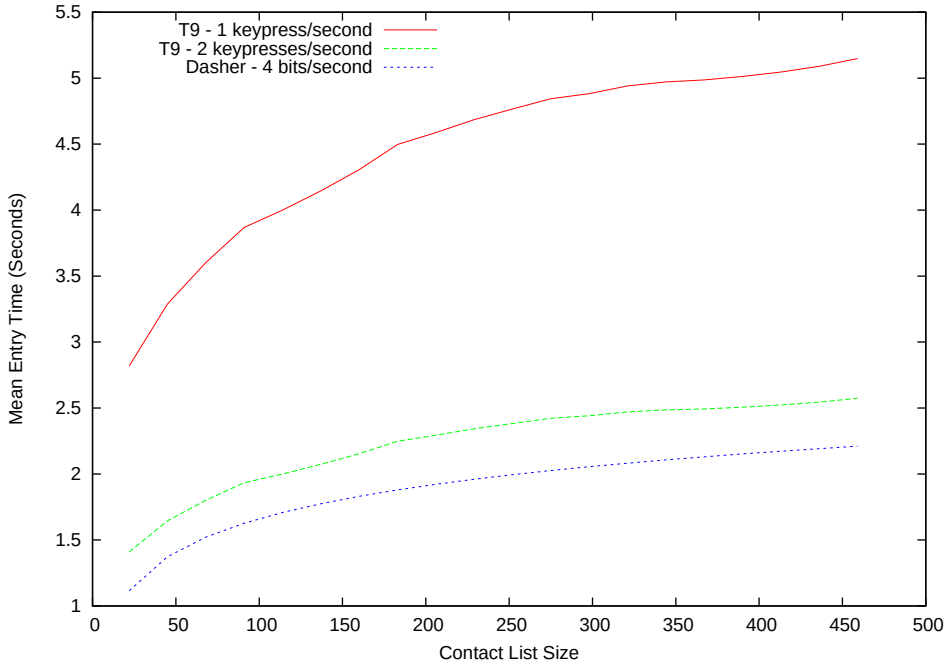


Figure 5.8: Average time required to unambiguously select entries from a contacts list. Values are plotted for contact lists sizes ranging from 22 to 460 entries. Rates are plotted assuming either 1 or 2 key presses per second for T9, or 4 bits per second for Dasher.

For Dasher however, the encoding scheme is dynamic, which allows for the possibility of adaptation. In particular, rather than assuming a uniform distribution over entries in the contacts list, any distribution may be used, for example based on the number of times that each entry has been retrieved in the past. Contacts which are accessed often would therefore be easier to recall. Adding a uniform Dirichlet prior with parameter α, this gives

$$P(x) = \frac{n(x) + \frac{\alpha}{q}}{N + \alpha} \tag{5.2}$$

where $n(x)$ is the number of times that entry x has been retrieved, $N = \sum_x n(x)$ is the total number of retrievals and q is the size of the contacts list. By associating a count with each node in the trie, predictions can be easily made using this model.

The net effect is that the entropy quoted in the previous section is an upper bound, and in practice it may be possible to significantly improve selection rates.

5.4 Further work

The analysis presented in this section has been theoretical in nature. The real utility of a novel user interface can only be determined by experiments conducted on human subjects using it for real tasks. It would therefore be appropriate to conduct such a test on the two interfaces described in this chapter. One quantity which would be revealing would be the

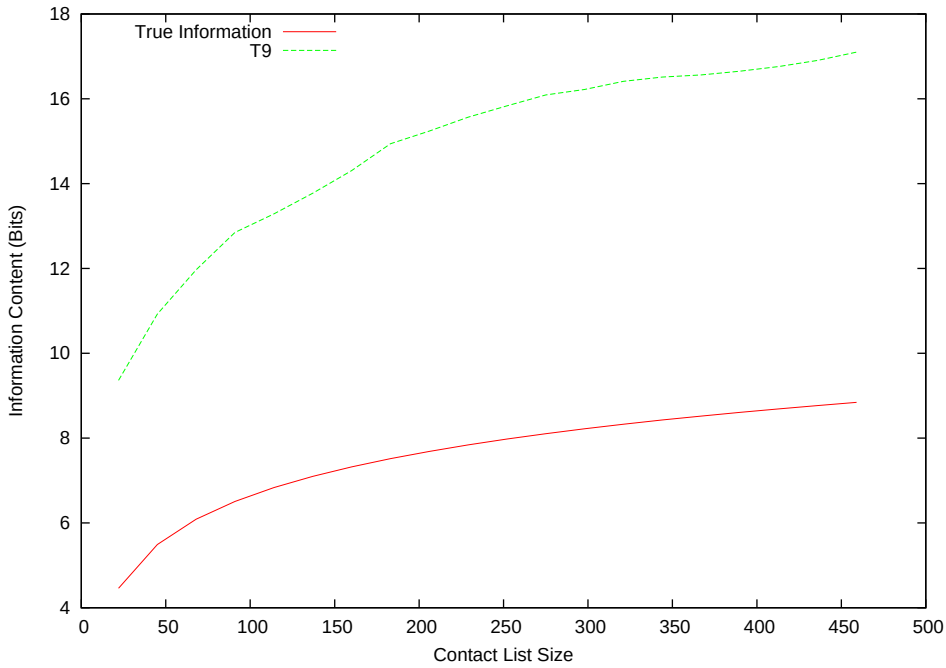


Figure 5.9: Average potential information content of keypresses required to select entries from the contact list, compared to the true information content assuming a uniform distribution over contacts. The potential information represents the number of bits which could be entered with the same number of keypresses were an optimal coding scheme to be used rather than T9.

time taken for users to perform tasks, such as looking up contacts using Dasher.

5.5 Conclusions

This chapter has presented an application for the Dasher user interface as a way of selecting items from an index, with the specific tasks of searching for substrings within a text document and selecting database records being considered. In the former case, a prototype application was developed to demonstrate the feasibility of the task, however, in its most basic form the method is quite demanding in terms of storage requirements. By making a trade-off between storage and computation at run time, it was shown that the storage requirements could be significantly reduced.

In the latter case, Dasher was compared to T9, a popular disambiguating text entry system used on mobile telephones. It was shown that Dasher provides a large improvement in terms of information efficiency, which can be further increased by using an adaptive interface whereby more probable entries become easier to select.

Chapter 6

Electronic Ink Analysis

6.1 Introduction

This thesis has so far only considered plain text documents, modelled as simple strings of tokens. No use has been made of richer forms of information which may be available, such as the layout of the text or multimedia elements. This final chapter makes a departure from this theme to consider the task of interpreting documents containing hand drawn electronic ink diagrams. These diagrams are typically entered using a stylus directly onto the screen of the device. Devices which are capable of this kind of interaction are becoming increasingly widely available, including Tablet PCs, Personal Digital Assistants (PDAs) and high end portable telephones.

Examples of the sort of diagram considered in this chapter are shown in Figure 6.1. The particular task which is of interest here is that of grouping fragments of ink strokes into perceptually meaningful objects (in other words, groups which would be recognised as an object by a human), and to label those groups as belonging to a particular object class. The approach will be illustrated on organisational charts such as those shown in Figure 6.1, where the object classes will be containers, which are box-like objects representing members of the organisation, and connectors, which are lines representing the relationships between members. As the diagram is created by direct digitisation of the stylus movement, there is no need to reconstruct ink strokes from a bitmap image as the data is already available. Furthermore, the temporal ordering of the strokes is available, which will be shown to be useful in the interpretation process.

A number of other attempts have been made to extract perceptually relevant objects from drawings [82] [83] [54], although these methods have not in general used a principled probabilistic framework. The work presented here was done with the assistance of Dr Martin Szummer, who provided the raw data, the definition of the feature set and library code used to perform basic manipulation of the input data and the resulting undirected graphs. Material in this chapter was presented in [90] and [23].

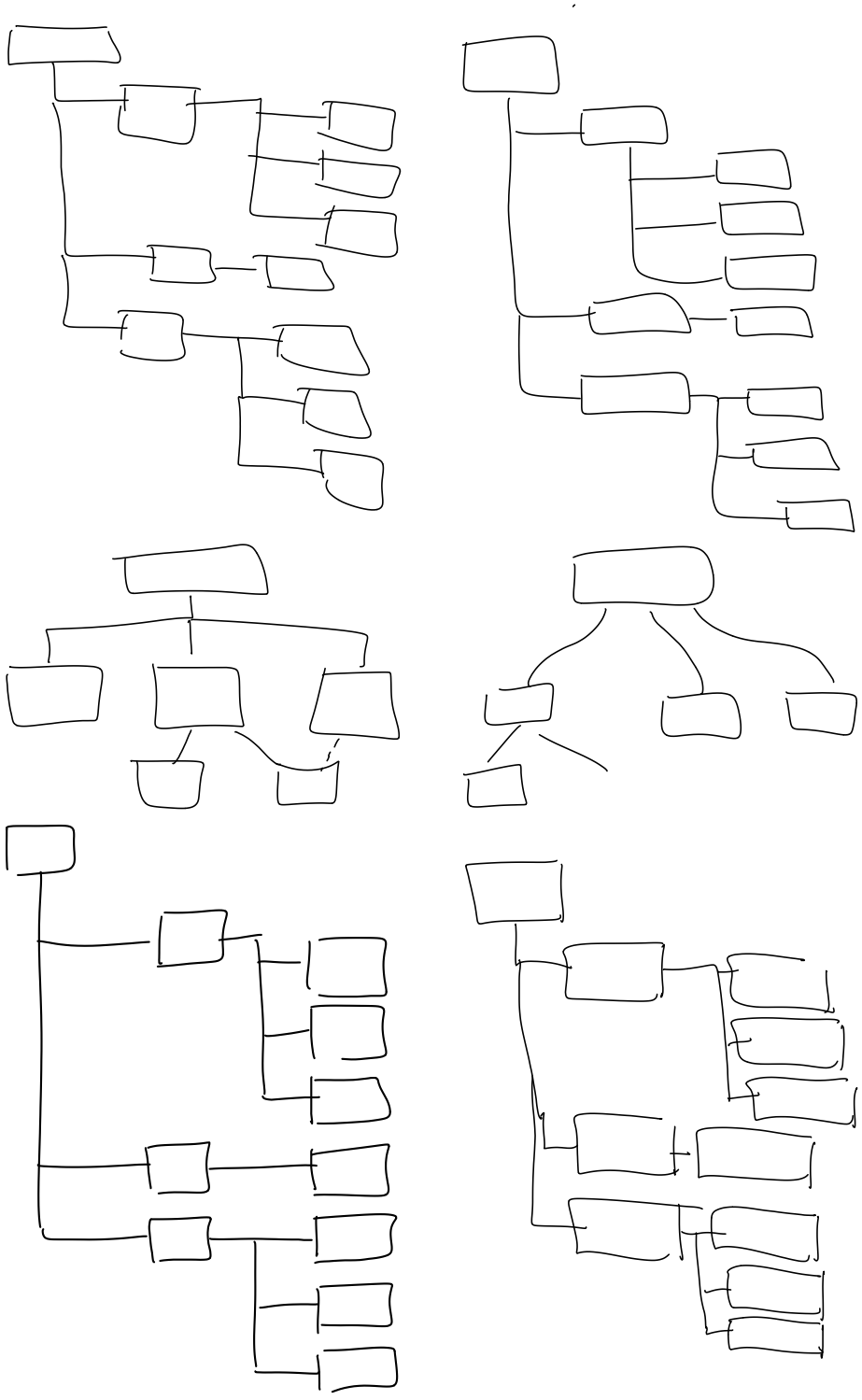
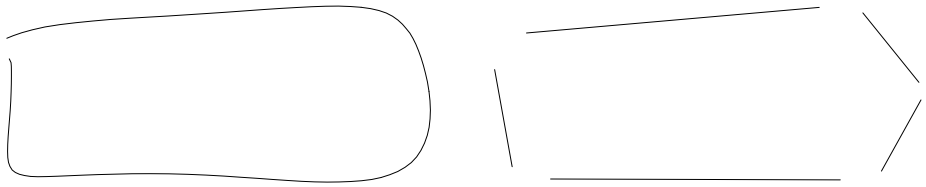
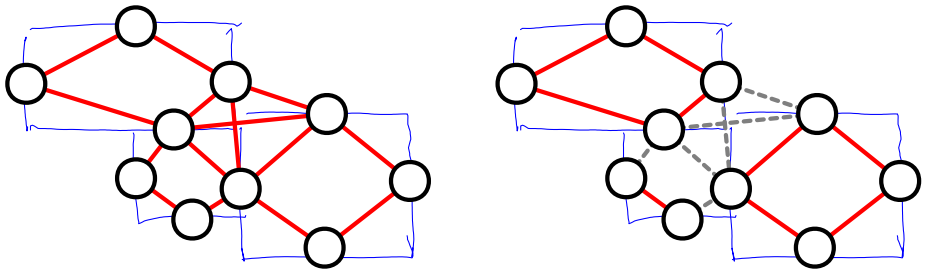


Figure 6.1: Six examples of ink diagrams used as input to the interpretation system.



(a) (b)

Figure 6.2: Division of the original input data (a) into fragments (b). The division is done on the basis that fragments must be straight to within a given tolerance.



(a) (b)

Figure 6.3: (a) An example of an undirected graph constructed from the input data in which each vertex represents an ink fragment. (b) Partition of the graph representing objects in the diagram. Dashed grey lines represent edges connecting vertices which are in different parts.

6.2 Overview of the algorithm

The input data is a set of ink strokes, which may span multiple objects. The first stage of the interpretation process is to split the strokes into fragments, which are assumed to belong to a single object. Fragments are defined to be sections of strokes which are straight to within a given tolerance (see Figure 6.2).

Having fragmented the strokes, an undirected graph, $\mathcal{G}$, containing one vertex for each ink fragment is constructed (See Figure 6.3(a)). Precise details of the construction of $\mathcal{G}$ will be given in Section 6.7.1. The goal of the algorithm will be to partition this graph so that each part contains the vertices corresponding to ink fragments which make up a single object, and to label each part according to the corresponding object class (See Figure 6.3(b)). The approach which will be taken is to construct a probabilistic model over labelled partitions conditioned on observed ink data.

6.3 Undirected graphical models

Multivariate probability distributions often contain internal structure, which can be exploited in order to make computation more efficient. Any representation which elucidates this structure is therefore desirable. The search for such representations has a long history, and has generated methods such as Bayesian networks [38], undirected graphical models [51] and factor graphs [45].

The most relevant of these representations is the undirected graphical model. Consider a distribution over three variables, y_1, y_2 and y_3, denoted $P(y_1, y_2, y_3)$. In the most general case, the distribution is specified by a function of all three variables. In some cases however, it may be possible to represent the distribution as a product of two functions (referred to as *potentials*), one of y_1 and y_2 only, and the other of y_2 and y_3 only. In other words

$$P(y_1, y_2, y_3) = \frac{1}{Z}\psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3) \tag{6.1}$$

where Z is a normalising constant,

$$Z = \sum_{y_1, y_2, y_3} \psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3) \tag{6.2}$$

Now suppose that y_2 is observed. In this case the conditional distribution can be written

$$P(y_1, y_3 \mid y_2) = \frac{1}{Z'}\psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3) \tag{6.3}$$

where

$$Z' = \sum_{y_1, y_3} \psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3) \tag{6.4}$$

Conditioning further on y_3,

$$\begin{aligned} P(y_1 \mid y_2, y_3) &= \frac{P(y_1, y_3 \mid y_2)}{P(y_3 \mid y_2)} && (6.5)\\ &= \frac{\psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3)}{\sum_{y_1} \psi_{12}(y_1, y_2) \cdot \psi_{23}(y_2, y_3)} && (6.6)\\ &= \frac{\psi_{23}(y_2, y_3) \cdot \psi_{12}(y_1, y_2)}{\psi_{23}(y_2, y_3) \cdot \sum_{y_1} \psi_{12}(y_1, y_2)} && (6.7)\\ &= \frac{\psi_{12}(y_1, y_2)}{\sum_{y_1} \psi_{12}(y_1, y_2)} && (6.8) \end{aligned}$$

The conditional distribution is independent of y_3. In other words, y_1 and y_3 are *conditionally independent* [25] given y_2, written $y_1 \perp\!\!\!\perp y_3 \mid y_2$.

This structure can be represented by constructing a graph consisting of a vertex for each variable and an edge connecting the vertices corresponding to any pair of variables which appear together in a potential in the full distribution. Viewed another way, there is one potential in the distribution for each clique, or fully connected subgraph in the graph, so the distribution corresponding to an arbitrary graph can be written as

$$P(\{x_i\}) = \frac{1}{Z}\prod_{j \in \mathcal{C}} \psi_j\left(\{x_i\}_j\right) \tag{6.9}$$

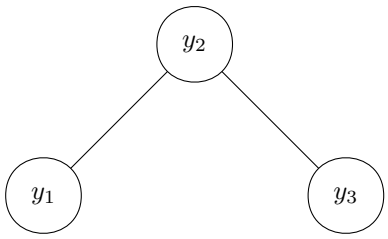


Figure 6.4: The undirected graphical model corresponding to the probability distribution given in (6.1). Vertices correspond to the variables over which the distribution is defined, and cliques, which in this case are (y_1, y_2) and (y_2, y_3), correspond to the potentials.

Where $\mathcal{C}$ is the set of cliques and the notation $\{x_i\}_j$ is used to represent the set of vertices in clique j. The graph corresponding to (6.1) is shown in Figure 6.4. In general, two variables y_i and y_j are conditionally independent given a subset of the variables if the subset *separates* y_i and y_j on the graph, or equivalently, if all paths between y_i and y_j on the graph pass through a member of the subset. This result has powerful implications in terms of the efficiency of computation using the model [51], a topic which will be returned to in detail in a later section.

Moral graphs [38] derived from Bayesian networks, constitute a special case of the undirected graphical model, where the potentials have a simple interpretation as the conditional probability tables of each variable given its parents.

6.3.1 Conditional random fields

Conditional random fields are a particular type of undirected graphical model used when a conditional distribution is required based on observed data [48]. Let $\boldsymbol{x}$ be a vector representing the observations, and $\{y_i\}$ be the set of variables over which the distribution is to be defined. As in Section 6.3, when conditioned on $\boldsymbol{x}$, the model has an independence structure represented by an undirected graph whose nodes are variables in $\{y_i\}$. To produce such a distribution, we use potentials which depend on *features* of the observed data. Typically, these are binary features defined for all vertices and edges on the graph, denoted $f_{ik}^{(1)}(\boldsymbol{x})$ and $f_{ijk}^{(2)}(\boldsymbol{x})$ respectively. The model has the form

$$P(\{y_i\} \mid \boldsymbol{x}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \psi_i(y_i, \boldsymbol{x}) \prod_{ij \in \mathcal{E}} \psi_{ij}(y_i, y_j, \boldsymbol{x}) \tag{6.10}$$

where $\mathcal{V}$ and $\mathcal{E}$ are the vertices and edges of the graph respectively. The potentials are given by

$$\psi_i(y_i, \boldsymbol{x}) = \exp \sum_k f_{ik}^{(1)}(\boldsymbol{x}) w_k^{(1)}(y_i) \tag{6.11}$$

$$\psi_{ij}(y_i, y_j, \boldsymbol{x}) = \exp \sum_k f_{ijk}^{(2)}(\boldsymbol{x}) w_k^{(2)}(y_i, y_j) \tag{6.12}$$

where $w^{(1)}$ and $w^{(2)}$ are weights which are associated with the values of $\{\boldsymbol{y}\}$. In many cases binary features are used, taking the values $+1$ and -1. Roughly speaking, positive weights indicate that the corresponding labels are compatible with a particular feature and negative weights indicate incompatibility, with larger magnitudes increasing the strength of the effect. The features are dependent on the index of the clique to which they are applied, which allows systematic variation where there is a natural correspondance between the observed data and the variables over which the distribution is defined. For example, features which may be used in a natural language processing setting include "The word at position i is 'the'" and "Position i is in the first 3 words of a sentence". One of the key strengths of the conditional random field is the arbitrary way in which the distribution can depend on the data. There is no computational cost involved in extending the range of the data dependences beyond that associated with the computation of the feature vectors themselves. Note that the simple model presented here can be generalised in a straightforward way, for example to include features that span larger cliques, or that take a continuous value rather than being binary.

The original applications of conditional random fields were in tasks associated with natural language processing, such as part-of-speech tagging or shallow parsing [85]. However, the approach is general and may be applied to any graph structure for which inference in the corresponding undirected model is tractable. Conditional random fields, including a Bayesian extension of the original maximum likelihood approach, have previously been applied to the same data set as used below but for the simpler task of labelling ink fragments without reference to partitioning [68]. Other work [88] has extended the CRF to perform multiple inference tasks simultaneously but has not considered partitioning of non-chain graphs (graphs with a chain- or tree-structure generally result in relatively simple inference problems. See later for a discussion of computational requirements). For an overview of conditional random fields see [94].

6.3.2 Models over partitions

Before introducing the approach which will be used in the remainder of this chapter, it is instructive to consider another possibility. One simple way of constructing a model over partitions would be simply to view the process as a labelling task, assigning each vertex a label and then defining the parts to be contiguous regions with the same label. This approach yields a standard labelling problem, so a conditional random field would be well suited to the task. There are however a number of serious problems.

Firstly, it is difficult to know in advance how many labels are necessary. In order to make sure that any partition can be realised, the number of labels must be equal to the size of the largest clique in the graph. The computational complexity of the algorithm scales as L^C where L is the number of labels and C is the size of the largest clique in the graph. This complexity potentially limits the extent to which this approach can be applied to real-life problems (see Section 6.6 for a more detailed discussion of computational requirements). A second, and possibly more serious problem is that representation in terms of labels introduces

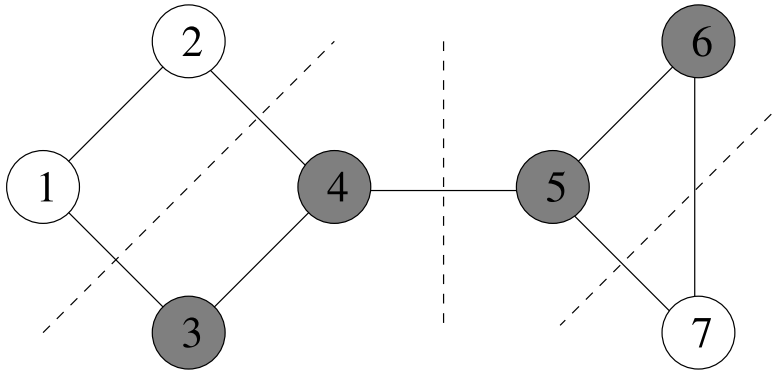


Figure 6.5: An example of a labelled partition. Vertices are partitioned as follows: $(1, 2, +)$, $(3, 4, -)$, $(5, 6, -)$, $(7, +)$, where the last symbol in each group indicates the label assigned to that part.

a degeneracy: the particular choice of labels is unimportant. In other words, the labels can be permuted without affecting the final partition. The number of ways of labelling a given partition depends on the number of parts present, so for example, if three mutually connected vertices are labelled with three labels, there are three assignments corresponding to the partition in which all vertices belong to the same part, but six assignments which correspond to the fully disjoint partition. If probability mass is assigned evenly amongst the possible ways of labelling the partition, it is possible that the disjoint case, even if it is more probable, might not be the single most probable arrangement, simply because the mass is spread more thinly.

To avoid these problems, the approach taken here is to extend the graphical model framework to work directly with labelled partitions of the graph. The following section develops these extensions from a theoretical viewpoint. Application to the specific problem of ink analysis will be left until later.

6.4 Theoretical development

Let $\mathcal{G}$ be an undirected graph as described in Section 6.2, consisting of vertices $\mathcal{V}$ and edges $\mathcal{E}$. Assume that $\mathcal{G}$ is triangulated, so that every cycle of length greater than three is spanned by a chord. Triangulation can always be achieved by adding edges, but usually at the expense of increasing the maximum clique size, and therefore computational complexity.

Let S be a partition of $\mathcal{G}$, that is, a set of non-empty subsets of $\mathcal{V}$, such that each vertex in $\mathcal{V}$ is a member of precisely one subset. Each subset is referred to as a *part* of $\mathcal{G}$. In the following, the term *partition* will always refer to a *contiguous* partition:

Definition 1. *A partition of $\mathcal{G}$ is **contiguous** if and only if all parts are internally connected. In other words, if i and j are vertices contained within the same part, there exists a path on $\mathcal{G}$ between i and j entirely contained within that part.*

A labelled partition of $\mathcal{G}$ is represented by $\mathcal{Y} = (\mathsf{S}, \boldsymbol{y})$, where S describes the partition and $\boldsymbol{y} \in \{-1, +1\}^M$ is a vector containing the labels associated with each part. For example, a partition of seven elements into four parts could be $\mathsf{S} = \{\{1,2\}\{3,4\}\{5,6\}\{7\}\}$, $\boldsymbol{y} = [+1, -1, -1, +1]$. This configuration is shown in Figure 6.5. Let $\mathbb{Y}$ be the set of all possible labelled partitions of $\mathcal{G}$. Note that M, the length of $\boldsymbol{y}$, depends on S. Let t_i be the index of the part to which vertex i is assigned, so that y_{t_i} is the label given to that vertex. The conditional probability distribution over $\mathbb{Y}$ has the form

$$P(\mathcal{Y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{i \in \mathcal{V}} \psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}) \prod_{i,j \in \mathcal{E}} \psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}), \tag{6.13}$$

where $\boldsymbol{x}$ is the observed data, $\boldsymbol{\theta}$ is a vector representing the model parameters collectively, and $Z(\boldsymbol{\theta})$ is a normalisation constant. $\psi_i^{(1)}$ are unary potentials defined for each vertex, and $\psi_{ij}^{(2)}$ are pairwise potentials defined for each edge. As for the standard CRF, the unary potentials introduce a data-dependent bias towards assigning one label or the other to each vertex. The pairwise potentials model the compatibility between the parts and labels of neighbouring vertices, and are also data dependent. These potentials depend on $\boldsymbol{x}$ through feature vectors, $\boldsymbol{g}_i$ and $\boldsymbol{f}_{ij}$, defined for each vertex i and edge (i,j) respectively. The potentials have the form

$$\psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}, \boldsymbol{\theta}) = \begin{cases} \phi(\boldsymbol{w}_+ \cdot \boldsymbol{g}_i(\boldsymbol{x})) & \text{if } y_{t_i} = +1 \\ \phi(\boldsymbol{w}_- \cdot \boldsymbol{g}_i(\boldsymbol{x})) & \text{if } y_{t_i} = -1 \end{cases}, \tag{6.14}$$

where $\phi(\cdot)$ is a non-linear mapping, and $\boldsymbol{w}_+$ and $\boldsymbol{w}_-$ are vectors of feature weights depending on the label of the appropriate vertex. For now, an exponential non-linearity, $\phi : x \mapsto \exp(x)$ will be used, although in general other functions may be substituted (for example, the probit function). The pairwise potentials are defined by

$$\psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}, \boldsymbol{\theta}) = \begin{cases} \phi\left(\boldsymbol{v}_{\mathrm{ss}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } t_i = t_j,\ y_{t_i} = y_{t_j} \\ \phi\left(\boldsymbol{v}_{\mathrm{sd}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } t_i \neq t_j,\ y_{t_i} = y_{t_j} \\ \phi\left(\boldsymbol{v}_{\mathrm{dd}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } t_i \neq t_j,\ y_{t_i} \neq y_{t_j} \end{cases} \tag{6.15}$$

where $\boldsymbol{v}_{\mathrm{ss}}$, $\boldsymbol{v}_{\mathrm{sd}}$ and $\boldsymbol{v}_{\mathrm{dd}}$ are vectors of feature weights to be used when i and j belong to the same part, different parts with the same label, and different parts with different labels respectively. The fourth case, corresponding to vertices with different labels in the same part, does not occur by definition. The parameters in $\boldsymbol{\theta}$ are therefore $(\boldsymbol{w}_+, \boldsymbol{w}_-, \boldsymbol{v}_{\mathrm{ss}}, \boldsymbol{v}_{\mathrm{sd}}, \boldsymbol{v}_{\mathrm{dd}})$. As adding a constant to all weights which correspond to a particular feature simply results in multiplication of the potentials by a constant, there is a redundancy in the weight vectors. In practice, $\boldsymbol{w}_-$ and $\boldsymbol{v}_{\mathrm{dd}}$ are therefore constrained to be $\mathbf{0}$.

6.4.1 Training

The overall goal of the model above is to provide labelled partitions of unseen data. Before this task can be performed however, it is necessary to estimate the model parameters, $\boldsymbol{\theta}$. These parameters are learned from example data. Given a labelled training example,$(\boldsymbol{x}, \mathcal{Y})$, the posterior probability of the parameters is given by Bayes' rule,

$$P(\boldsymbol{\theta} \mid \boldsymbol{x}, \mathcal{Y}) \propto P(\mathcal{Y} \mid \boldsymbol{x}, \boldsymbol{\theta}) \cdot P(\boldsymbol{\theta}), \tag{6.16}$$

where $P(\boldsymbol{\theta})$ is a prior distribution over the weights. The model is trained by finding the maximum *a posteriori* (MAP) weights using a quasi-Newton gradient ascent algorithm (specifically, the BFGS implementation contained in the MATLAB statistics toolbox). A significant advantage is that the model is convex in the parameters, meaning that a gradient ascent algorithm will always find the global maximum. Substituting the details of the distribution into (6.16) gives a log posterior, $\mathcal{L}$, of

$$\mathcal{L} = \sum_i \log \psi_i^{(1)} + \sum_{ij} \log \psi_{ij}^{(2)} - \log Z + \log\left(\mathrm{P}(\boldsymbol{\theta})\right) \tag{6.17}$$

Taking the gradient with respect to a parameter θ_k therefore gives

$$\frac{\partial}{\partial \theta_k}\mathcal{L} = \sum_{i \in \mathcal{V}} \frac{\partial}{\partial \theta_k} \log \psi_i^{(1)} + \sum_{ij \in \mathcal{E}} \frac{\partial}{\partial \theta_k} \log \psi_{ij}^{(2)} - \frac{1}{Z}\frac{\partial Z}{\partial \theta_k} + \frac{\partial}{\partial \theta_k} \log\left(\mathrm{P}(\boldsymbol{\theta})\right) \tag{6.18}$$

The partition function, Z, is given by

$$Z(\boldsymbol{\theta}) = \sum_{\mathcal{Y}} \prod_{i \in \mathcal{V}} \psi_i^{(1)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}) \prod_{i,j \in \mathcal{E}} \psi_{ij}^{(2)}(\mathcal{Y}, \boldsymbol{x}; \boldsymbol{\theta}). \tag{6.19}$$

So, if the θ_k is a parameter of the unary potentials, the gradient is given by

$$\begin{aligned}
\frac{1}{Z}\frac{\partial Z}{\partial \theta_k} &= \frac{1}{Z}\frac{\partial}{\partial \theta_k} \sum_{\mathcal{Y}} \prod_{i \in \mathcal{V}} \psi_i^{(1)} \prod_{ij \in \mathcal{E}} \psi_{ij}^{(2)} && (6.20)\\
&= \frac{1}{Z} \sum_{\mathcal{Y}} \prod_{i \in \mathcal{V}} \psi_i^{(1)} \prod_{ij \in \mathcal{E}} \psi_{ij}^{(2)} \sum_{m \in \mathcal{V}} \frac{1}{\psi_m^{(1)}} \frac{\partial}{\partial \theta_k} \psi_m^{(1)} && (6.21)\\
&= \sum_{m \in \mathcal{V}} \sum_{\mathcal{Y}} P(\mathcal{Y}) \frac{\partial}{\partial \theta_k} \log \psi_m^{(1)} && (6.22)\\
&= \sum_{m \in \mathcal{V}} \left\langle \frac{\partial}{\partial \theta_k} \log \psi_m^{(1)} \right\rangle && (6.23)
\end{aligned}$$

where explicit functional dependencies have been dropped for clarity. The brackets, $\langle \cdots \rangle$, represent expectation with respect to the distribution over $\mathbb{Y}$ given by the current parameter

values. Combining (6.23) with (6.18) gives

$$\frac{\partial}{\partial\theta_k}\mathcal{L} = \sum_{i\in\mathcal{V}}\left(\frac{\partial}{\partial\theta_k}\log\psi_i^{(1)} - \left\langle\frac{\partial}{\partial\theta_k}\log\psi_i^{(1)}\right\rangle\right) + \frac{\partial}{\partial\theta_k}\log\left(\mathrm{P}\left(\boldsymbol{\theta}\right)\right) \tag{6.24}$$

For the particular case of exponential non-linearities, this expression simply reduces to

$$\frac{\partial}{\partial\theta_k}\mathcal{L} = \sum_{i\in\mathcal{V}}\left(g_{ik}\delta_{y_{t_i},y_k} - \left\langle g_{ik}\delta_{y_{t_i},y_k}\right\rangle\right) + \frac{\partial}{\partial\theta_k}\log\left(\mathrm{P}\left(\boldsymbol{\theta}\right)\right) \tag{6.25}$$

where δ_{y_i,y_k} takes the value 1 if y_k, the label associated with weight k is equal to y_{t_i}, the actual label assigned to vertex i, and zero otherwise. The gradients with respect to parameters of the pairwise potentials have a similar form. If multiple training examples are provided then optimisation is performed on the joint probability of all of the data. As the algorithm works in log space, optimising the joint probability simply results in a summation over all training examples,

$$\frac{\partial}{\partial\theta_k}\mathcal{L} = \sum_j\sum_{i\in\mathcal{V}}\left(g_{ik}\delta_{y_i^{(j)},y_k} - \left\langle g_{ik}\delta_{y_i,y_k}\right\rangle\right) + \frac{\partial}{\partial\theta_k}\log\left(\mathrm{P}\left(\boldsymbol{\theta}\right)\right) \tag{6.26}$$

where the superscript (j) indicates the value in the j^{th} training example. This expression is closely related to the training rule for Boltzmann machines with hidden units (see Chapter 43 in [56] for further details).

The expectation in (6.26) requires the computation of marginal probability distributions for individual vertices and pairs of vertices connected by an edge. Furthermore, the optimisation algorithm needs to evaluate (6.13) explicitly, which in turn requires evaluation of the partition function. Both of these tasks involve summations over subsets of possible labelled partitions. This summation can be performed efficiently by message passing using a modified version of the sum-product algorithm. The details of this algorithm will be given in Section 6.5.

6.4.2 Inference

Having trained the model on example data, we next usually use the model to assign a labelled partition to previously unseen data. The approach taken is to return the most probable configuration,

$$\mathcal{Y}^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}}\prod_{i\in\mathcal{V}}\psi_i^{(1)}\left(\mathcal{Y},\boldsymbol{x};\boldsymbol{\theta}\right)\prod_{i,j\in\mathcal{E}}\psi_{ij}^{(2)}\left(\mathcal{Y},\boldsymbol{x};\boldsymbol{\theta}\right). \tag{6.27}$$

This process is similar to the computation required to find the partition function, Z, and can be performed using the max-product in much the same way. This algorithm will also be described in Section 6.5.

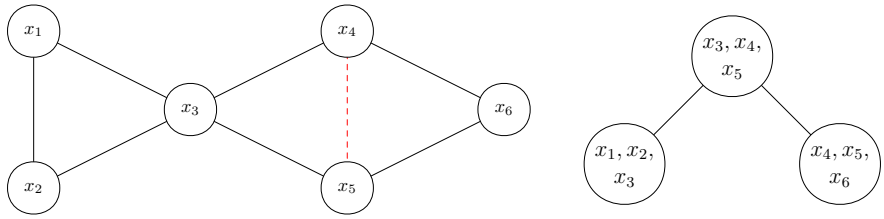


Figure 6.6: An example of a undirected graph (left) and the corresponding junction tree (right). The original graph, shown in black, has a cycle of length four, so an additional edge, shown in red, must be added to perform triangulation.

6.5 Operations over labelled partitions

6.5.1 Message passing

Efficient inference in undirected graphical models can be performed systematically by message passing algorithms. These algorithms reduce the task to local computations involving subsets of the vertices, the results of which are passed as messages along the edges in the graph.

For exact inference two such algorithms are widely used: the sum-product algorithm which is used for marginalisation over subsets of the variables, and the max-product algorithm which is used to find the most probable configurations of the variables. Equivalently, the max-sum algorithm performs the same role when logarithms of the probabilities are available. As an illustration, consider the sum-product algorithm.

The goal of this algorithm is to compute the sum over a subset of the variables, resulting in the marginal distribution of those which remain. The first stage of the algorithm is to *triangulate* the graph, adding additional edges so that there are no loops of size four or greater which are not spanned by a chord. This process has no effect on the distribution, as the potentials associated with the added edges simply evaluate to one for all configurations, but is necessary to ensure that the algorithm terminates.

Having triangulated the graph, a *junction tree* is constructed. The junction tree is a second graph containing one node for each maximal clique in the original graph (a clique is a maximally connected subgraph). The vertices on the junction tree are connected with edges so that they possess the *running intersection property*. This requirement is satisfied if for all pairs of vertices on the graph corresponding to cliques sharing a common vertex, there exists at least one path between them such that all vertices on the path also correspond to cliques containing the common vertex. As long as the original graph is triangulated, a junction graph with this property always exists. Further more, the junction graph will always be a tree. However, it may not be unique, and selection between the possible trees may affect the computational requirements of the algorithm. An example of a graph and the associated junction tree is shown in Figure 6.6.

Consider the sum over all of the vertices in the graph, which can be written as

$$S = \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5}\sum_{x_6}\psi_{123}(x_1,x_2,x_3)\,\psi_{345}(x_3,x_4,x_5)\,\psi_{456}(x_4,x_5,x_6) \quad (6.28)$$

$$= \sum_{x_1}\sum_{x_2}\sum_{x_3}\psi_{123}(x_1,x_2,x_3)\sum_{x_4}\sum_{x_5}\psi_{345}(x_3,x_4,x_5)\sum_{x_6}\psi_{456}(x_4,x_5,x_6) \quad (6.29)$$

The summation over the third potential is clearly independent of x_1, x_2 and x_3. It is therefore not necessary to repeat this computation for all values of these variables. Instead, it can be pre-computed and stored as a function of x_4 and x_5. Conceptually, the result of this computation can be regarded as a message passed from clique (456) to clique (345).

To express the message passing framework more formally, let C_i represent the set of variables present in the i^{th} clique, which has potential ψ_i. Let S_{ij} be the separator between cliques i and j: the set of all variables appearing in both C_i and C_j. The message passed from i to j is then

$$\mu_{i\to j}(S_{ij}) = \sum_{C_i\setminus S_{ij}}\psi_i(C_i)\prod_{k\in\mathcal{N}(i)\setminus j}\mu_{k\to i}(S_{ik}) \quad (6.30)$$

in which $\mathcal{N}(i)$ is the neighbourhood of clique i on the junction graph. Assume that the set of variables over which summation is not to be performed is entirely contained within a single clique. This clique can be designated as the root of the junction tree, and messages are passed from the leaves of the tree to the root. Each node (excluding the root) can pass a message as soon as it has received incoming messages from all but one of its neighbours, with the outgoing message being passed to that which remains.

A new potential over the root clique, C_1, can now be defined as

$$\psi_1'(C_1) = \psi_1(C_1)\prod_{k\in\mathcal{N}(1)}\mu_{k\to 1}(C_1) \quad (6.31)$$

which is the un-normalised marginal distribution over configurations of that clique. Partial or full summation over these configurations can then be performed explicitly as required.

If marginal distributions over all cliques are required then performing the full message passing routine each time would result in wasteful duplication of calculations. A little thought reveals that a second, outward pass from the root back to the leaves will compute messages passed in both directions along all edges on the junction tree. The resulting messages are sufficient to compute all marginals at once.

The limiting factor in the computation is the size of the summations which must be performed. The largest summation will result from computing the outgoing messages from the largest clique, and involves L^C calculations, where C is the size of that clique and L is the number of values that each variable can take.

6.5.2 Message passing for labelled partitions

Message passing for traditional undirected graphical models is well established, but it cannot be applied directly to the model over labelled partitions; this can be understood intuitively as a result of the fact that a partition of the graph cannot be broken down into components localised to individual vertices. Fortunately, equivalent algorithms can be found, a formal derivation of which is presented below. The derivation presented here follows the conditions for the possibility of local computation provided by Shenoy and Shafer [87].

If G is a subset of $\mathcal{V}$, let $\mathcal{Y}_G \in \mathbb{Y}_G$ denote a labelled partition of the corresponding induced subgraph. *Consistency* is then defined as follows:

Definition 2. *Labelled partitions $\mathcal{Y}_G$ and $\mathcal{Y}_H$, of subgraphs G and H respectively, are **consistent**, denoted $\mathcal{Y}_H \backsim \mathcal{Y}_G$, if and only if:*

1. *All vertices appearing in $G \cap H$ are assigned the same label by $\mathcal{Y}_G$ and $\mathcal{Y}_H$, and*
2. *All pairs of vertices appearing in $G \cap H$ are in the same part in $\mathcal{Y}_G$ if and only if they are in the same part in $\mathcal{Y}_H$.*

The notation $\hat{\mathcal{Y}}_G(\mathcal{Y}_{G\cup H})$ is used to denote the unique labelled partition of G which is consistent with $\mathcal{Y}_{G\cup H}$. The maximal cliques of $\mathcal{G}$ are defined in the usual way, and are denoted $C_1, \ldots, C_N$. If b and t are two cliques, and b contains all vertices from t which appear in cliques other than t, then b is said to be a *branch* and t is the corresponding *twig*. For example, in Figure 6.6, the node (x_4, x_5, x_6) is a twig, and the node (x_3, x_4, x_5) is the corresponding branch.

Let ψ be a *valuation* on a subset of $\mathcal{V}$. In the case of standard belief propagation, valuations are functions assigning a real, non-negative value to possible configurations of subsets of the variables. Here however, a valuation on a subset G will be defined as a function mapping $\mathbb{Y}_G$ to the non-negative real numbers. $\mathbb{V}_G$ is the set of all valuations on G. In the case where the valuation is over the whole of $\mathcal{G}$, the range of the valuation will be interpreted as being proportional to the probability of the corresponding labelled partition. In the case of valuations defined over subsets of $\mathcal{V}$ the valuations are referred to as potentials of which those defined in (6.13) are an example. Two operations on valuations must be defined:

1. **Combination:** Suppose G and H are subsets of $\mathcal{V}$ and ψ_G and ψ_H are valuations on those subsets. The operation of combination defines a mapping $\otimes : \mathbb{V}_G \times \mathbb{V}_H \mapsto \mathbb{V}_{G\cup H}$, such that
$$\psi_G \otimes \psi_H(\mathcal{Y}_{G\cup H}) \triangleq \psi_G\left(\hat{\mathcal{Y}}_G(\mathcal{Y}_{G\cup H})\right) \cdot \psi_H\left(\hat{\mathcal{Y}}_H(\mathcal{Y}_{G\cup H})\right). \tag{6.32}$$
2. **Marginalisation:** Suppose G and H are subsets of $\mathcal{V}$ such that $G \subseteq H$, and ψ_G and ψ_H are valuations as before. Marginalisation is a mapping $\downarrow : \mathbb{V}_H \mapsto \mathbb{V}_G$ such that
$$\psi_H^{\downarrow G}(\mathcal{Y}_G) \triangleq \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_G} \psi_H(\mathcal{Y}_H). \tag{6.33}$$

A valuation over the whole graph is said to factor if it can be written as the combination of valuations on the cliques,

$$\psi\left(\mathcal{Y}\right) = \bigotimes_{i=1}^{N} \psi_i\left(\mathcal{Y}_i\right), \tag{6.34}$$

where i runs over the cliques in $\mathcal{G}$. As combination allows products of valuations over subsets of a clique to be written in terms of a single valuation over the whole clique, the model given in (6.13), excluding the partition function, is in this form. In order to proceed to develop efficient algorithms, it is first necessary to demonstrate that three axioms of Shenoy and Shafer are satisfied:

Axiom 1. Commutativity and associativity of combination. *If G, H and K are subsets of $\mathcal{V}$, for any valuations ψ_G, ψ_H and ψ_K, we have $\psi_G \otimes \psi_H = \psi_H \otimes \psi_G$ and $\psi_G \otimes (\psi_H \otimes \psi_K) = (\psi_G \otimes \psi_H) \otimes \psi_K$.*

Proof. Follows directly from the definition of combination. □

Axiom 2. Consonance of marginalisation, *If G, H and K are subsets of $\mathcal{V}$ such that $K \subseteq G \subseteq H$, for any valuations ψ_G, ψ_H and ψ_K,*

$$\left(\psi_H^{\downarrow G}\right)^{\downarrow K} = \psi_H^{\downarrow K}. \tag{6.35}$$

Proof. Writing the marginalisation explicitly,

$$\begin{aligned}\left(\psi_H^{\downarrow G}\right)^{\downarrow K} &= \sum_{\mathcal{Y}_G \backsim \mathcal{Y}_K} \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_G} \psi_H\left(\mathcal{Y}_H\right) \\ &= \sum_{\mathcal{Y}_H \backsim \mathcal{Y}_K} \psi_H\left(\mathcal{Y}_H\right) = \psi_H^{\downarrow K},\end{aligned} \tag{6.36}$$

where the second line follows as for any $\mathcal{Y}_H \backsim \mathcal{Y}_K$ there is a unique $\mathcal{Y}_G$ such that $\mathcal{Y}_G \backsim \mathcal{Y}_K$ and $\mathcal{Y}_H \backsim \mathcal{Y}_G$, and for any $\mathcal{Y}_H \not\backsim \mathcal{Y}_K$, no such $\mathcal{Y}_G$ exists. □

Axiom 3. Distributivity of marginalisation over combination, *If G and H are subsets of $\mathcal{V}$, for any valuations ψ_G and ψ_H, $(\psi_G \otimes \psi_H)^{\downarrow G} = \psi_G \otimes (\psi_H^{\downarrow G \cap H})$.*

Proof. Performing an explicit expansion gives

$$\begin{aligned}(\psi_G \otimes \psi_H)^{\downarrow G} &= \sum_{\mathcal{Y}_{G\cup H} \backsim \mathcal{Y}_G} \psi_G\left(\hat{\mathcal{Y}}_G\left(\mathcal{Y}_{G\cup H}\right)\right) \cdot \\ &\qquad \psi_H\left(\hat{\mathcal{Y}}_H\left(\mathcal{Y}_{G\cup H}\right)\right) \\ &= \psi_G\left(\mathcal{Y}_G\right) \cdot \sum_{\mathcal{Y}_{G\cup H} \backsim \mathcal{Y}_G} \psi_H\left(\hat{\mathcal{Y}}_H\left(\mathcal{Y}_{G\cup H}\right)\right) \\ &= \psi_G\left(\mathcal{Y}_G\right) \cdot \sum_{\mathcal{Y}_H \backsim \hat{\mathcal{Y}}_{G\cap H}(\mathcal{Y}_G)} \psi_H\left(\mathcal{Y}_H\right),\end{aligned} \tag{6.37}$$

which is equal to $\psi_G \otimes (\psi_H^{\downarrow G\cap H})$ by definition. □

6.5.3 Sum-product algorithm

Having shown that Shenoy and Shaffer's axioms are satisfied, the next two sections formally develop an extension of the sum-product algorithm suitable for probability distributions over partitions. As with the more usual form of this algorithm, the resulting method exploits the known structure of $\mathcal{G}$ by passing messages containing the results of local computations. The goal of the algorithm is to compute sums over a subset of all possible partitions, such as those needed for the partition function, as given in (6.19). This task should be contrasted with that of the usual sum-product algorithm [51], which sums over assignments of labels to the vertices. Since the summation is over a different domain it will be necessary to modify the messages passed and the ranges of summation. Later, in Section 6.5.5, the max-product algorithm for labelled partitions will be considered.

Suppose the cliques of $\mathcal{G}$ are numbered $C_1, \ldots, C_N$, such that C_1 is a clique whose marginal distribution we wish to compute. In other words, we wish to find the sum:

$$f_s(\mathcal{Y}_1) = \sum_{\mathcal{Y} \backsim \mathcal{Y}_1} P^*(\mathcal{Y}) = (P^*(\mathcal{Y}))^{\downarrow C_1}, \tag{6.38}$$

where $\mathcal{Y}_1$ represents a local labeled partition of clique C_1 and P^* is a (possibly unnormalised) probability distribution over labelled partitions of $\mathcal{G}$. Furthermore, suppose that the numbering of the cliques is such that for all k, C_k is a twig in the graph $C_1 \cup C_2 \cup \ldots \cup C_k$. Such an ordering is always possible if $\mathcal{G}$ is triangulated. According to Axiom 2, this can be expressed as

$$\begin{aligned} f_s(\mathcal{Y}_1) &= \left((P^*(\mathcal{Y}))^{\downarrow \mathcal{V}\setminus C_N} \right)^{\downarrow C_1} \\ &= \left(\left(\bigotimes_{i=1}^{N} \psi_i(\mathcal{Y}_i) \right)^{\downarrow \mathcal{V}\setminus C_N} \right)^{\downarrow C_1} \\ &= \left(\left(\bigotimes_{i=1}^{N-1} \psi_i(\mathcal{Y}_i) \right) \otimes \left(\psi_N(\mathcal{Y}_N)^{\downarrow C_N \cap \mathcal{V}} \right) \right)^{\downarrow C_1}. \end{aligned} \tag{6.39}$$

In the last step, Axiom 3 has been used. C_N is a twig by construction. Let C_B be a corresponding branch, then $C_N \cap \mathcal{V} = C_N \cap C_B$, hence

$$f_s(\mathcal{Y}_1) = \left(\left(\bigotimes_{\substack{i=1 \\ i\neq B}}^{N-1} \psi_i(\mathcal{Y}_i) \right) \otimes \psi_B(\mathcal{Y}_B) \otimes \left(\psi_N(\mathcal{Y}_N)^{\downarrow C_N \cap C_B} \right) \right)^{\downarrow C_1}. \tag{6.40}$$

In other words, the problem can be converted to an equivalent marginalisation over a graph with one less clique in which the potential for C_B has been replaced according to:

$$\psi_B \leftarrow \psi_B \otimes \left(\psi_N^{\downarrow C_N \cap C_B} \right). \tag{6.41}$$

By repeatedly eliminating cliques in this way we can systematically remove cliques until only C_1 remains. Any further summation which is required (either to give marginals over a smaller subset of vertices, or to calculate the partition function) can be performed explicitly.

6.5.4 Message passing

The result of the elimination illustrated in (6.41) can be interpreted in terms of a message passed from C_N to the rest of the graph. Messages are passed between cliques along edges in a *junction tree* equivalent to that described in Section 6.5.1. Let $\mu_{i\to j}(\mathcal{Y}_j)$ be the message passed from C_i to C_j. The form of the message is a list of labelled partitions of the intersection $C_i \cap C_j$, each of which has an associated scalar value. The messages are updated iteratively according to the rule:

$$\mu_{i\to j}(\mathcal{Y}_j) \leftarrow \sum_{\mathcal{Y}_i \backsim \mathcal{Y}_j} \psi_i(\mathcal{Y}_i) \prod_{\substack{k\in\mathcal{N}(i)\\ k\neq j}} \mu_{k\to i}(\mathcal{Y}_i), \tag{6.42}$$

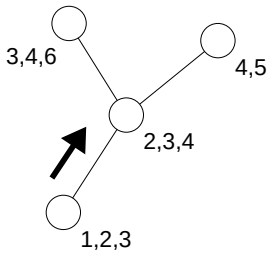
with the outgoing messages from a clique being updated once all incoming messages from the other neighbouring cliques $\mathcal{N}(\cdot)$ have been received. Note that the domain of the incoming messages is only a subset of the vertices in clique i, so the messages must be extended to fully cover $\mathcal{Y}_i$. This is done by simply assigning to each configuration of the clique the value of the unique consistent configuration of the separator. As the junction tree has no cycles, this process will terminate after a finite number of iterations. Having updated all of the messages, it is then possible to find f_s using

$$f_s(\mathcal{Y}_1) = \psi_1(\mathcal{Y}_1) \prod_{k\in\mathcal{N}(1)} \mu_{k\to 1}(\mathcal{Y}_1). \tag{6.43}$$

Having defined the algorithm formally, it is useful to also give an intuitive interpretation. The message passed from C_i to C_j can be interpreted as a statement summarising the values of the 'upstream' potentials for labelled partitions which are consistent with each labelled partition of the separator between C_i and C_j. See Figure 6.7 for an example of the message passing process. As is the case with the usual form of the sum-product algorithm, the same messages are used in computing different marginals. Marginal distributions for all cliques can be found simultaneously with a single bidirectional pass of the message update rule.

6.5.5 Max-product algorithm

Just as is the case for the usual form of the sum-product algorithm, it is possible to replace the summation in (6.38) with a maximisation to obtain the max-product algorithm. This replacement is equivalent to a redefinition of marginalisation to represent the maximum valuation consistent with the sub-partition rather than the sum over all valuations. This algorithm is used to compute maximisations, for example the configuration of C_1 in the most probable



(a)

Potential	Value
$\psi_{ij}\,(t_i = t_j, y_i = y_j)$	0.6
$\psi_{ij}\,(t_i \neq t_j, y_i = y_j)$	0.4
$\psi_{ij}\,(t_i \neq t_j, y_i \neq y_j)$	0.2

(b)

Partition	Label	Value
(123)	+	0.216
(123)	-	0.216
(12)(3)	+, +	0.096
(12)(3)	+, -	0.024
⋮	⋮	⋮
(1)(2)(3)	-,-,-	0.064

(c)

Partition	Label	Value
(23)	+	0.336
(23)	-	0.336
⋮	⋮	⋮
(2)(3)	-,-	0.272

(d)

Figure 6.7: An example of message passing. (a) The junction tree corresponding to $\mathcal{G}$. (b) The potentials, in this case uniform and independent of data for clarity. (c) The clique potential for the clique consisting of vertices 1, 2 and 3. (d) The message passed from (123) to (234), concerning labelled partitions of vertices 2 and 3.

labelled partition,

$$\mathcal{Y}_1^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}_1} \max_{\mathcal{Y} \backsim \mathcal{Y}_1} P^*\left(\mathcal{Y}\right). \tag{6.44}$$

In the context of probabilistic inference, such a maximisation is necessary when searching for the most probable configuration. Message passing is done in the same way as described above, with a modified message update rule.

$$\mu_{i \to j}\left(\mathcal{Y}_j\right) \leftarrow \max_{\mathcal{Y}_i \backsim \mathcal{Y}_j} \psi_i\left(\mathcal{Y}_i\right) \prod_{\substack{k \in \mathcal{N}(i) \\ k \neq j}} \mu_{k \to i}\left(\mathcal{Y}_i\right). \tag{6.45}$$

Having updated all of the messages, $\mathcal{Y}_1^{\mathrm{MAX}}$ can be found using

$$\mathcal{Y}_1^{\mathrm{MAX}} = \arg\max_{\mathcal{Y}_1} \psi_1\left(\mathcal{Y}_1\right) \prod_{k \in \mathcal{N}(1)} \mu_{k \to 1}\left(\mathcal{Y}_1\right). \tag{6.46}$$

To find the global maximum configuration, this method can be repeated for all possible roots, and the global partition can be reconstructed as the union of the local configurations (which will be consistent with one another).

Again, it is instructive to consider the intuitive meaning of the messages. In this case they can be interpreted as statements about the maximum value that can be achieved 'upstream' as a function of the clique separator configuration. When the next cluster computes its maximum configuration, the contribution of downstream potentials can therefore be incorporated from the messages rather than having to be recomputed from scratch each time.

6.6 Complexity and the edge-dual representation

Section 6.3.2 presented an alternative approach to labelled partitions which was shown to be too demanding in terms of computational requirements for real-life applications. Yet another alternative representation which avoids these problems is to use indicator variables, $\check{\boldsymbol{x}}\left(\mathcal{Y}\right)$,

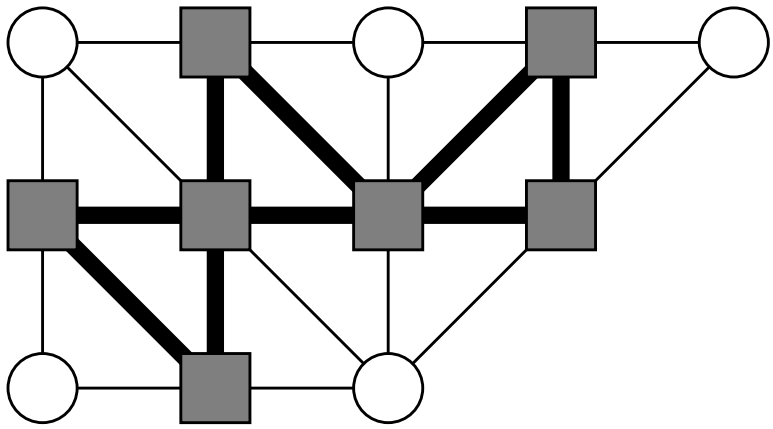


Figure 6.8: An example of an undirected graph (circular vertices and light lines) and the corresponding edge-dual graph (square vertices and heavy lines).

for each edge in $\mathcal{G}$. For binary labels, these variables are over six possible values: two states corresponding to segments belonging to the same part with each label, and four corresponding to different parts with all four combinations of labels. To construct a graphical model for these variables, we define the *edge-dual graph*:

Definition 3. *For any graph $\mathcal{G}$, the **edge-dual graph**, $\check{\mathcal{G}} = \left(\check{\mathcal{V}}, \check{\mathcal{E}}\right)$ contains one vertex for each edge in $\mathcal{G}$. Vertices in $\check{\mathcal{G}}$ are connected by an edge if and only if all vertices connected to their corresponding edges in $\mathcal{G}$ belong to the same clique.*

An example of an edge-dual graph is shown in Figure 6.8. Every labelled partition of $\mathcal{G}$ corresponds to a unique configuration of the edge-dual vertices, but there are configurations of the edge-dual vertices which do not correspond to labelled partitions. Hence,

Definition 4. *A configuration of the edge-dual vertices is **valid** if and only if it corresponds to a labelled partition of $\mathcal{G}$.*

Invalid configurations arise when pairwise constraints yield contradictory information; following one path between two vertices on $\mathcal{G}$ indicates that they are in the same part, whereas another path indicates that they are not, or their labels disagree. It is possible to establish the validity of a configuration using only calculations local to cliques on $\check{\mathcal{G}}$.

Suppose $P^*\left(\check{\boldsymbol{x}}\left(\mathcal{Y}\right)\right)$ is a probability distribution over labelled partitions of $\mathcal{G}$ as represented by the edge-dual variables. As before, consider the task of summing P^* over all partitions in order to find the partition function. Rather than expressing the summation in terms of partitions, we can work directly with $\check{\boldsymbol{x}}$, provided that the summation is limited to those configurations which are valid. This constraint can be achieved by introducing an indicator function, $\mathbb{I}\left(\check{\boldsymbol{x}}\right)$, which takes the value 1 if $\check{\boldsymbol{x}}$ is valid and 0 otherwise,

$$\sum_{\mathcal{Y}} P^*\left(\check{\boldsymbol{x}}\left(\mathcal{Y}\right)\right) = \sum_{\check{\boldsymbol{x}}} \mathbb{I}\left(\check{\boldsymbol{x}}\right) \cdot P^*\left(\check{\boldsymbol{x}}\right). \tag{6.47}$$

	Clique Size					
	2	3	4	5	6	n
(a)	2	5	15	52	203	Bell no. B_n
(b)	*6*	*22*	*94*	*454*	*2430*	A001861 [100]
(c)	6	216	46656	6.0×10^7	4.7×10^{11}	$6^{n(n-1)/2}$
(d)	16	216	4096	1.0×10^5	3.0×10^6	$(2n)^n$

Table 6.1: Sizes of the configuration tables for each of the methods. (a) Unlabelled Partitions (these are the Bell numbers). (b) Binary labelled partitions (c) Binary labelled edge-dual representation. (d) Binary labelled part IDs (lower bound).

There is a one-to-one correspondence between cliques in $\mathcal{G}$ and $\check{\mathcal{G}}$, so functions which factor according to $\mathcal{G}$ also factor according to $\check{\mathcal{G}}$. If P^* factors, we can write

$$\sum_{\mathcal{Y}} P^* \left(\check{\boldsymbol{x}} \left(\mathcal{Y} \right) \right) = \sum_{\check{\boldsymbol{x}}} \left(\prod_i \mathbb{I}_i \left(\check{\boldsymbol{x}}_i \right) \cdot \check{\psi}_i \left(\check{\boldsymbol{x}}_i \right) \right), \tag{6.48}$$

where i ranges over the cliques of $\check{\mathcal{G}}$. In (6.48), the local nature of $\mathbb{I}$ has been used to factor it as well as P^*. The result is a sum over a function which factors according to $\check{\mathcal{G}}$, so it can be found using the standard sum-product algorithm.

As there is a one-to-one correspondence between valid edge-dual configurations, and labelled partitions of $\mathcal{G}$, this algorithm is in many respects equivalent to that presented in Section 6.5.3. However, in two important respects it is less efficient. Firstly, as the sum includes edge-dual configurations which are invalid, the number of terms in the sum is significantly greater. Secondly, it is necessary to determine the validity of the current configuration for each term, which introduces additional overhead. The algorithm presented in Section 6.5.3 may be regarded as an efficient implementation of this algorithm, where the validity of configurations is pre-computed, and only those which are valid are included in the sum.

6.6.1 Complexity

The dominant factor in the complexity of the message passing algorithm is the time taken to process all possible partitions of the largest clique. Table 6.1 lists the number of possible configurations for the various cases, which are shown graphically in Figure 6.9. It can be seen from the table that the method described in Section 6.5 offers a considerable improvement in the complexity of the calculations.

6.7 Implementation details

Having derived a probabilistic model over labelled partitions of an undirected graph, and shown that efficient inference is possible, the remainder of this section returns to the intended application of electronic ink analysis.

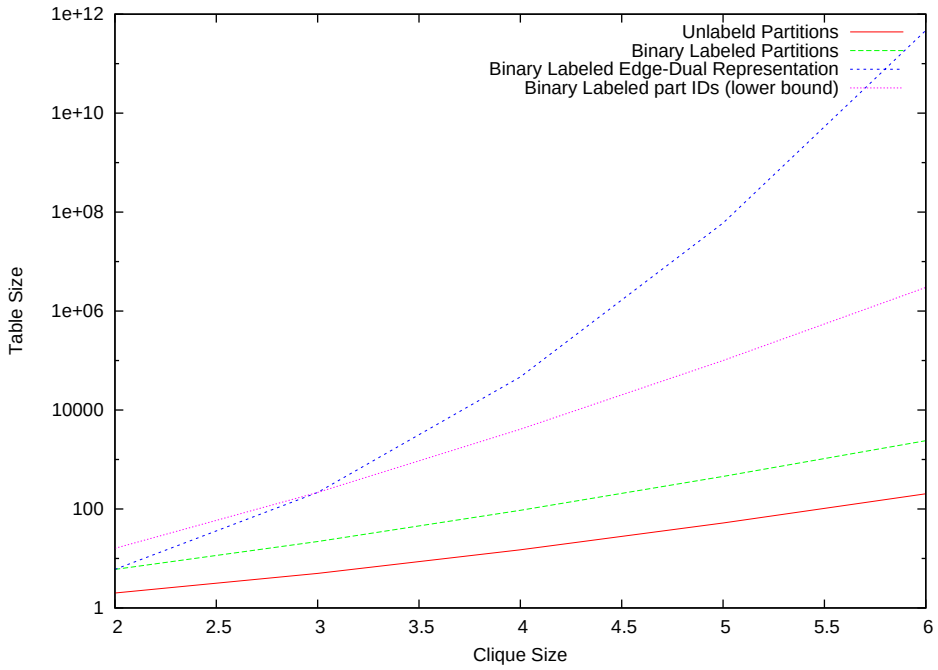


Figure 6.9: Sizes of the message tables for each of the methods.

6.7.1 Graph construction

The graph which is partitioned, $\mathcal{G}$, is constructed by first building a candidate graph by connecting all pairs of fragments which are closer than a preset threshold distance. This graph is not necessarily triangulated, so additional edges are added as necessary. Pairwise feature vectors are generated for all edges on the new graph, including those which were added during triangulation, as there is no significant computational cost involved in doing so, and these features might provide useful information. In fact, the whole process can be viewed as an approximation to the fully connected graph, so these additional features actually improve the faithfulness of the approximation.

6.7.2 Feature set

We chose features to reflect the spatial and temporal distribution of ink strokes, for example lengths and angles of fragments, whether two fragments were drawn with a single stroke, and the temporal ordering of strokes. In addition, a number of 'template' features are used, which were designed to capture important higher level aspects of the ink, such as the presence of T-junctions.

As some of the features correspond to real-valued quantities such as lengths and angles, some care needs to be taken. One approach would be to use the quantity directly. However, features of this kind would enforce a linear relationship between the value of the quantity and the strength of the weight, which may not be appropriate. Instead, we divided the range of

Name	Feature Count	Type
Length	11	Histogram
Angle	9	Histogram
Neighbour Angle	9	Histogram
Neighbour Relative Angle	11	Histogram
Neighbour Distance	11	Histogram
Long Fragment	1	Indicator
Full Box	1	Template
T-Junction Count	2	Count
Neighbour Full Box	1	Template
Right/Bottom	1	Template
Box	3	Template
Bias	1	Constant

Table 6.2: Unary features

Name	Feature Count	Type
Angle	11	Histogram
Distance	11	Histogram
Temporal Ordering	6	Histogram
T-Junction	1	Template
Box Full	1	Template
Right	1	Template
Bottom	1	Template
neighbour_two_box	1	Template
Similar Direction	1	template
Aligned	1	Template
Right Angle	1	Template
Bias	1	Constant

Table 6.3: Pairwise features

the feature value into subintervals, each of which was associated with a binary valued feature. Collectively, these groups are referred to as 'histogram features'.

We used a total of 61 unary features and 37 pairwise features were used, including bias features which take the value 1 for all potentials and provide a way of adjusting the overall preference for labels as well as the 'natural' degree of segmentation. Descriptions of the features are given in Tables 6.2 and 6.3.

6.7.3 Priors

We used Gaussian priors for all of the weights. In the case of histogram features the Gaussian was correlated so that the weights of intervals corresponding to similar bins were encouraged

to take on similar values. More formally, the covariance matrix, S had the form

$$S_{ij} = w_s \exp\left(-\frac{(i-j)^2}{w_r}\right) \tag{6.49}$$

which may be viewed as an approximation to a continuous mapping from real valued features to weights, with a Gaussian process prior. For the experiments described below, we used values of 2.0 for w_s and w_r, as well as for the variance on the independent priors for the non-histogram features.

6.8 Experimental evaluation

To test the performance of the method we used a database of 40 example diagrams, consisting of a total of 2157 ink fragments. We generated three random splits, each consisting of 20 examples used for training and 20 used for evaluation. We trained the model by finding the MAP weights as described in Section 6.4.1. The model was then tested by finding the most probable partition and labelling as described in Section 6.4.2, and counting errors made against ground-truth data. The undirected graphs constructed for the example diagrams had a mean tree-width of 4.0.

For comparison, we also evaluated two related models. These models performed labelling only, without considering partitioning, and both are implemented as standard CRFs. The first of these models includes unary potentials which are dependent on the class of the vertex, and uses pairwise potentials given by

$$\psi_{ij}^{(2)}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \begin{cases} \phi\left(\boldsymbol{v}_{\mathrm{s}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } y_i = y_j \\ \phi\left(\boldsymbol{v}_{\mathrm{d}} \cdot \boldsymbol{f}_{ij}(\boldsymbol{x})\right) & \text{if } y_i \neq y_j \end{cases}, \tag{6.50}$$

where $\boldsymbol{v}_{\mathrm{s}}$ and $\boldsymbol{v}_{\mathrm{d}}$ are weights corresponding to vertices i and j having the same and different labels respectively. The second related model does not use pairwise potentials at all — ink fragments are labelled independently of the other labellings. In the following, the full model performing labelling and partitioning is referred to as model PLI. LI is the model performing labelling only with pairwise potentials, and L is the model with unary potentials only.

6.8.1 Learned weights

Figures 6.10 and 6.11 show the weights which were learned for each of the features. The majority of the features play some role in discriminating between labelled partitions, with a few features having particularly large weights. A number of the template features come near the top of the lists, particularly the T-junction template. This result illustrates the power of the CRF framework in allowing such arbitrary dependence on the observed data. Also note that one of the temporal features plays an important role in deciding whether fragments are

Model	Split			Mean
	1	2	3	
L	8.2%	9.3%	7.9%	$8.5\% \pm 0.4\%$
LI	4.6%	6.8%	1.9%	$4.4\% \pm 1.4\%$
% Δ LI/L	−44%	−27%	−76%	$-49\% \pm 14\%$
PLI	3.1%	3.8%	0.8%	$2.6\% \pm 0.9\%$
% Δ PLI/LI	−33%	−44%	−58%	$-45\% \pm 7\%$

Table 6.4: Labelling errors for the three models. Results are the mean of three cross-validation splits. Relative differences are shown between models L and LI, and between LI and PLI. The mean relative differences are aggregations of the differences for each split, rather than the differences between the means for individual models in order to reduce the effect of systematic variation between splits.

in the same object or not, indicating that the additional information available as a result of working directly with digitised ink strokes is valuable.

The need for histogram features can be seen by considering the length features as an example of the learned weights, shown in Figure 6.12. While intermediate lengths provide evidence for a container, both long and short fragments are likely to belong to connectors (the short fragments resulting from the decomposition of curves into short line fragments). This relationship could not be achieved using the implied linear mapping produced by using the feature value directly.

6.8.2 Error rates

Labelling error rates were measured as the fraction of the fragments which were incorrectly labelled, and are shown in Table 6.4 and Figure 6.13.

Evaluation of the performance in partitioning presented a more significant challenge, partially as there is not a standard metric for this measurement, and partially because of the lack of a comparable algorithm with which to compare the results. The metric which was chosen is a modified version of the error metric used by Martin *et. al.* [58],

$$E = \sum_i \frac{|R(S_1, i) \setminus R(S_2, i)| + |R(S_2, i) \setminus R(S_1, i)|}{|R(S_1, i)|} \tag{6.51}$$

where $R(S, i)$ is the set of vertices in the same part as vertex i in partition S. S_1 is the ground truth partition and S_2 is the partition returned by the algorithm. The mean grouping error using this metric was 36. Figure 6.14 shows the output of the algorithm on an example diagram.

6.9 Discussion

The results given in Section 6.8 show that the present approach is capable of providing high-quality labelled partitions. The data also illustrate an important point: simultaneous labelling and partitioning produces a significant improvement in labelling performance. This is easily understandable — the constraint that vertices within the same part must be labelled identically provides strong evidence for the labelling part of the algorithm, and the boundaries between regions of different labels are strong candidates for part boundaries. Hence the two aspects of the algorithm reinforce each other.

6.10 Future extensions

6.10.1 Additional features

There are a number of additional aspects of the data which cannot easily be taken into account using the method described above. In particular, properties of individual parts essentially produce features which depend on the partition itself. Examples of such features include information containing the relative closure of the ink strokes making up a part, or the convexity defined as the ratio of the area of the part to its convex hull. These features are likely to be highly informative in the application to electronic diagram analysis.

One possible approach to this problem is to use Monte-Carlo techniques, which will be discussed below. Alternatively, it may be possible to produce lists of the top n most probable configurations using a model without these per-part features, and then explicitly evaluate the probabilities under the more complex model in order to re-rank them.

6.10.2 Tree-width constraints

It has already been shown that the time taken to perform inference is strongly dependent on the tree-width of the constructed graph. Ideally the fully connected graph should be used, as no matter how weak the interaction between two vertices there is no harm in taking it into account. Any deletion of edges from this graph should therefore be viewed as an approximation, and there is a trade-off between computational complexity and performance which can be varied by adjusting the graph which is used.

The method described above used a heuristic based on the spatial distance between ink strokes to construct the graph. While this was observed to give good results in practice, it may be worthwhile using a more sophisticated approach to refine the model further. Such an approach would attempt to construct a graph which is close to optimal in terms of the complexity/accuracy tradeoff.

While it is clearly not possible to know the results of the inference in advance, it is possible to estimate the importance of an edge from the local potential. By assigning a weight to each potential edge using this information, the goal would be to construct the graph with the

largest total edge weight given a constrained tree-width, with the constraint being set by the resources available to do inference.

Restricting the tree-width in this way is of course a non-trivial task and in practice it would most probably be necessary to resort to an approximate method. A simple rule would be a greedy algorithm, which could proceed either by starting with the fully disconnected graph and adding the most heavily weighted edges subject to the tree-width bound not being exceeded. Alternatively, the algorithm could start with a fully connected graph and delete edges.

6.10.3 Approximate inference

Monte Carlo methods

Monte Carlo methods provide a way of sampling from otherwise intractable probability distributions. By using such a method it is possible to sample from the set of labelled partitions of a previously unseen graph using an already trained model. One technique which is particularly appropriate for this model is that of Swendsen–Wang cuts [89]. This method would permit sampling from partitions of a much more densely connected graph than is tractable using exact inference.

While not exactly representing the marginal distribution, the pairwise potentials provide a relatively informative estimate of the relationship between neighbouring vertices. At each update step Swendsen–Wang cuts use this information to propose updates to the labelled partition, in the form of merges and splits of partitions and changes to the labels of existing parts. These proposals can then be accepted stochastically using a Metropolis–Hastings like update rule. This method has previously been applied to image segmentation [3], which is in many respects very similar to ink analysis.

Other approximations

An alternative to using a Monte Carlo algorithm is to introduce an approximation to the full distribution which is designed to be as close as possible but permit efficient inference.

One such approach is loopy belief propagation [59]. In traditional problems this is performed by using the same update rule as ordinary belief propagation, but operating on untriangulated graphs. In this case the junction graph is not a tree, and messages can potentially propagate infinitely around loops, which invalidates the proofs of exact inference provided for the non-loopy case. However, in many cases the process converges and good performance is achieved.

In the case of partitioning, there is however an additional problem, namely that partitions are only permitted which are consistent. Consider the case of a large ring of vertices in which most of the edges give strong evidence that all vertices should be assigned to the same part, with the exception of a single edge which has strong evidence that its vertices belong to different parts. In this situation the graph is frustrated - going around the loop there would

be a preference for a single boundary, which is not consistent. The exact inference method effectively introduces an additional potential for the whole ring enforcing consistency, as is illustrated by the edge dual representation, but constraining things in this way is not possible without triangulating the ring. It is not clear how a loopy algorithm would behave in such a situation, so further investigation is needed.

An alternative approach would be to use an approximation based on a reduced tree width graph. Consistency could be enforced on the approximate graph. Such an approach might be based on variational inference, where an approximate distribution is selected from a permitted family to minimise the KL divergence between the approximation and the exact distribution. A further option is the use of expectation propagation, which optimises the approximate distribution by matching moments.

6.11 Conclusions

This chapter has presented a probabilistic model over labelled partitions of an undirected graph, and has shown that the structure of the graph may be used to efficiently perform exact inference with message passing algorithms. An application of the model to the task of parsing hand-drawn diagrams has been demonstrated. Experimental results illustrate that it is possible to obtain high-quality results using this technique. The results obtained prove that in the present application, labelling accuracy is improved by performing partitioning at the same time.

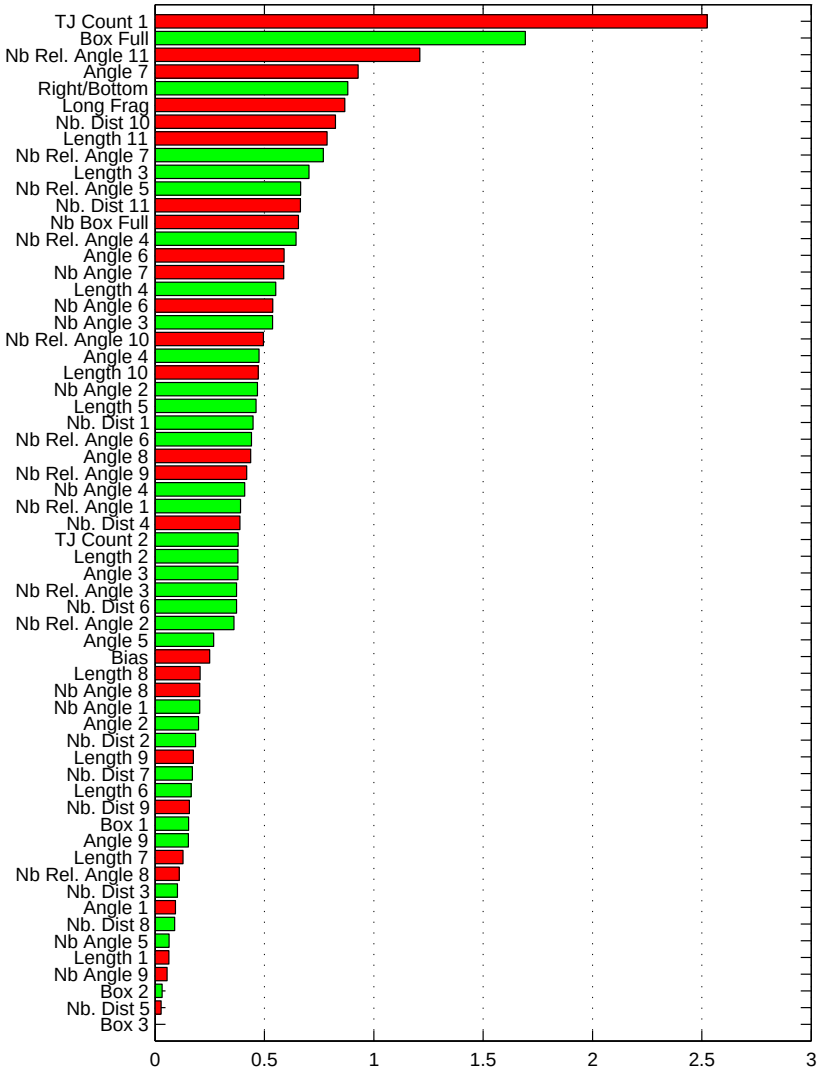


Figure 6.10: Learned unary weights. Positive values are shown in green and indicate a preferences for containers. Negative weights are shown in red and indicate a preference for connectors.

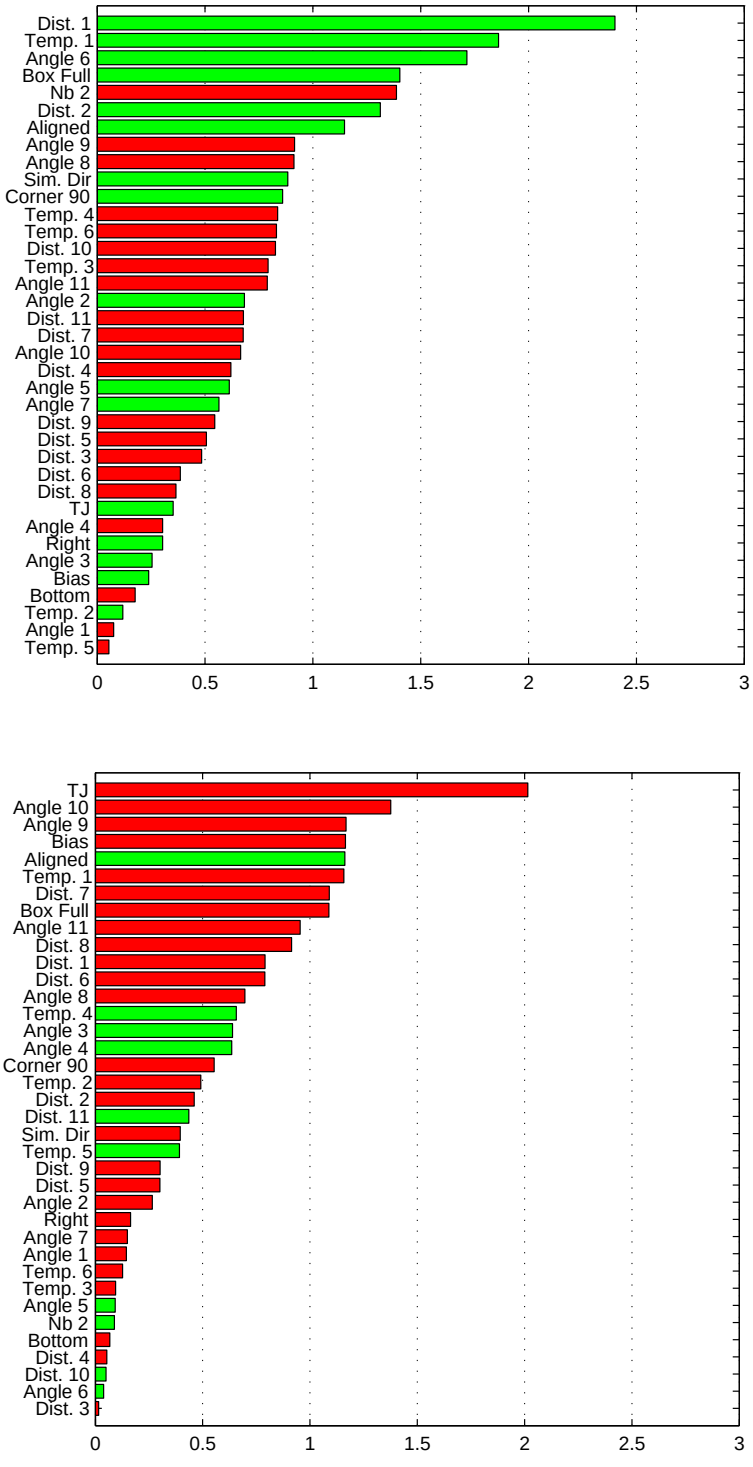


Figure 6.11: Pairwise weights for same object/same label (top) and different object/same label (bottom) configurations. Positive weights are shown in green and favour the named configuration. Negative weights are shown in red.

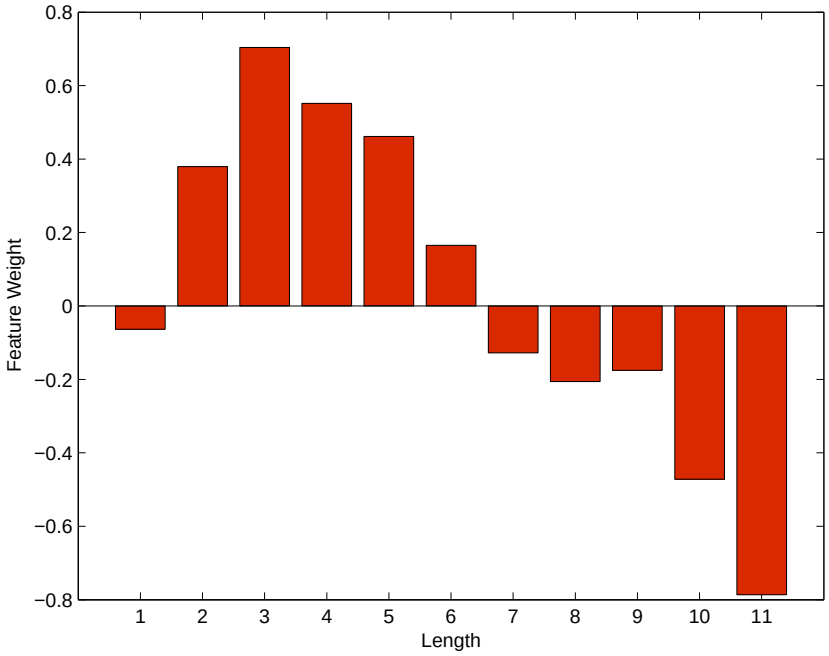


Figure 6.12: Feature for the unary length features. Positive weights correspond to a preference for categorisation as a container rather than a connector.

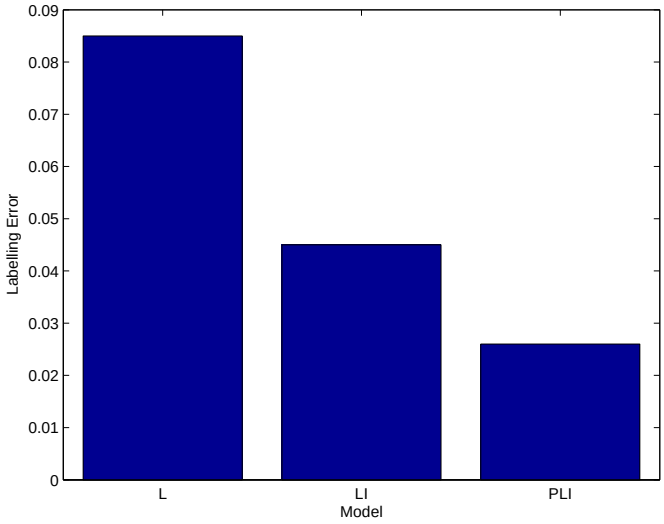


Figure 6.13: Mean labelling errors for the three models.

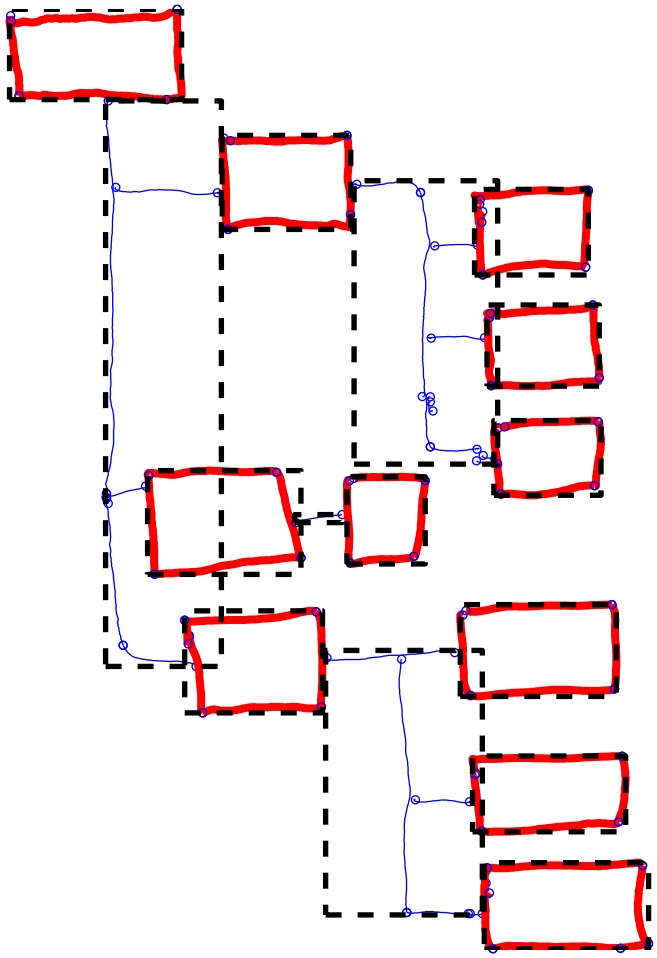


Figure 6.14: Example labellings and groupings: the most probable partition and labelling using model PLI. Heavy lines indicate fragments which have been classified as containers and lighter lines indicate connectors. Groups of fragments which belong to the same part are outlined using a dashed box.

Chapter 7

Conclusions

Probabilistic language modelling is central to a wide variety of applications, from data compression to speech recognition. The field has a long history and much progress has been made over the years. However, this progress has been in many cases due to heuristic approaches which while improving predictive performance, do not necessarily increase our understanding of the underlying processes.

One of the key contributions of this thesis has been to demonstrate the relationship between an important existing approach, generalised PPM-A, and the hierarchical Dirichlet language model, which is a simple Bayesian model based on a clear theoretical framework. Although exact inference in the latter is not possible on a realistic time scale, this thesis has shown that generalised PPM-A is an excellent approximation to this model under realistic conditions. Many applications of language modelling, for example speech recognition, require real-time computation, whereas others such as information retrieval must process many thousands of documents within a time-frame which is acceptable to the user. These applications therefore present considerable restrictions on the available computational resources, making low cost approximations such as generalised PPM-A invaluable.

Using generalised PPM-A as a starting point, it was demonstrated that basic symbol-level language modelling can be extended to make use of higher level structure present in the text, namely the knowledge that the text stream consists of a sequence of separate word and non-word characters. This information permits the construction of a model which is able to make use of a word list, even if relative frequencies of the terms are not known.

This approach was shown to be beneficial when either no training text is available or the quantity of training text is very small, a result which in itself is potentially beneficial. However, the model presented in Chapter 3 did not perform as well as a simple symbol-level language model when large quantities of training text were available. To some extent this result is disappointing, as one might have expected a more sophisticated model with access to the same training data to be able to perform at least as well. This area can therefore be identified as one where there is potential for further investigation, with the aim of improving on the results presented here.

Chapter 4 investigated the application of hierarchical Dirichlet models to information retrieval. This approach was shown to perform favourably when compared to other models, giving an improvement in performance in at least some cases. However, perhaps more importantly, it was shown that the resulting method naturally includes document frequency information as a way to identify the most discriminative terms in the query. This information has long been used in information retrieval, but until now has either been missing or introduced in an *ad hoc* way in those approaches which have made use of language models. The natural inclusion of this information further demonstrates that much progress can be made using simple, but principled, probabilistic methods.

There is much scope for further work in the application of language modelling to information retrieval. Chapter 4 demonstrated one direction for further investigation by extending the hierarchical model to model passages within the documents. The resulting method was able to consider not only terms occurring within each passage, but also within the context of the document of which the passage is a part. This approach gave promising results when compared to the simpler method of treating passages as separate documents. However, it is likely that further progress can be made, for example by investigating alternate definitions of passages or looking at the possibility of a continuously varying measure of relevance throughout the document. Further examples of areas which could be considered are the use of 'topic' information in retrieval, and the use of information on word correlations within documents. In both of these areas the development of simple approximations is likely to be important.

Having looked at probabilistic models in the context information retrieval, Chapter 5 looked at the Dasher interface as a way of extending the use of probabilistic information to the process of entering search queries and to displaying the results. A prototype implementation of an interactive search tool for locating substrings in a text document showed that this approach is viable. A further investigation of this method as a means of locating contact information indicated that, at least in terms of a theoretical analysis, substantial improvements to the efficiency of the search process are possible. To complete this investigation, it is now necessary to evaluate the performance of these applications in trials on real users.

The final chapter in this thesis considered richer documents than those containing just plain text. In particular, the chapter focused on the task of interpreting hand-drawn electronic ink diagrams, which are entered using a pen-based interface. This model was shown to perform well on example data, and in particular it was shown that the use of partitioning information was able to improve labelling performance over models which considered labelling only.

There are a number of further applications of the model developed for diagram analysis, for example in segmenting bitmap images or video. It is likely that these applications will involve more complex inference tasks, so development of approximate methods will be required. Possible approaches to this problem have already been considered in Chapter 6.

Appendix A

Data Sets

A number of data sets were used for experimental evaluation purposes in this thesis. The section summarises the data contained in each.

A.1 Language modelling

A.1.1 Enron e-mail corpus

This dataset consists of approximately 500,000 e-mail messages sent by approximately 150 members of the staff of the Enron corporation, and was originally released as part of a legal investigation of the company. Preprocessing has been done to remove attachments [43].

A.1.2 Canterbury corpus

The Canterbury corpus [104] [1] aims to provide a standard data set for comparison of compression algorithms. The corpus consists of 11 files of varying time, summarised in Table A.1. Analysis in this thesis concentrates on the file `alice29.txt` as an example of natural English.

File	Abbreviation	Category	Size
alice29.txt	text	English text	152089
asyoulik.txt	play	Shakespeare	125179
cp.html	html	HTML source	24603
fields.c	Csrc	C source	11150
grammar.lsp	list	LISP source	3721
kennedy.xls	Excl	Excel Spreadsheet	1029744
lcet10.txt	tech	Technical writing	426754
plrabn12.txt	poem	Poetry	481861
ptt5	fax	CCITT test set	513216
sum	SPRC	SPARC Executable	38240
xargs.1	man	GNU manual page	4227

Table A.1: Files making up the Canterbury corpus.

A.2 Information retrieval

A.2.1 TREC corpus

The TREC corpus is a large data set consisting of articles taken from a variety of newswire and other sources. The data set is the basis of the TREC information retrieval competition [113]. The data is supplied on five CD-ROMs, although only discs 4 and 5 are used in the evaluations presented above. This data set consists of 528,155 documents spanning a total of 165,363,765 terms from a vocabulary of size 629,469 (after stop words removal).

Also provided are a number of query strings consisting of three parts, a title, description and narrative. Ground truth judgements are available concerning whether or not each of the documents is relevant to each of the queries. Two sets of fifty queries (referred to as TREC-7 and TREC-8) were used in this thesis.

A.2.2 Cranfield corpus

The Cranfield corpus is a relatively small information retrieval corpus consisting of 1400 abstracts on aeronautical engineering topics. The documents contain a total of 136935 terms from a vocabulary of size 4,632 (after stop word removal).

The Cranfield corpus also contains a set of 225 query strings with ground truth relevance judgements. Relevances are assigned on four levels, but for purposes of comparison to the TREC data set this was converted to a binary value for the experiments presented in this thesis.

Appendix B

Stop Word List

The following list of 319 stop words was used in the information retrieval experiments. It was made available by the Information Retrieval Group at the University of Glasgow [103].

a	anyone	both	eleven	four	however
about	anything	bottom	else	from	hundred
above	anyway	but	elsewhere	front	i
across	anywhere	by	empty	full	ie
after	are	call	enough	further	if
afterwards	around	can	etc	get	in
again	as	cannot	even	give	inc
against	at	cant	ever	go	indeed
all	back	co	every	had	interest
almost	be	computer	everyone	has	into
alone	became	con	everything	hasnt	is
along	because	could	everywhere	have	it
already	become	couldnt	except	he	its
also	becomes	cry	few	hence	itself
although	becoming	de	fifteen	her	keep
always	been	describe	fify	here	last
am	before	detail	fill	hereafter	latter
among	beforehand	do	find	hereby	latterly
amongst	behind	done	fire	herein	least
amoungst	being	down	first	hereupon	less
amount	below	due	five	hers	ltd
an	beside	during	for	herself	made
and	besides	each	former	him	many
another	between	eg	formerly	himself	may
any	beyond	eight	forty	his	me
anyhow	bill	either	found	how	meanwhile

might
mill
mine
more
moreover
most
mostly
move
much
must
my
myself
name
namely
neither
never
nevertheless
next
nine
no
nobody
none
noone
nor
not
nothing
now
nowhere
of
off
often
on
once
one
only
onto
or
other
others
otherwise
our
ours
ourselves
out
over
own
part
per
perhaps
please
put
rather
re
same
see
seem
seemed
seeming
seems
serious
several
she
should
show
side
since
sincere
six
sixty
so
some
somehow
someone
something
sometime
sometimes
somewhere
still
such
system
take
ten
than
that
the
their
them
themselves
then
thence
there
thereafter
thereby
therefore
therein
thereupon
these
they
thick
thin
third
this
those
though
three
through
throughout
thru
thus
to
together too
top
toward
towards
twelve
twenty
two
un
under
until
up
upon
us
very
via
was
we
well
were
what
whatever
when
whence
whenever
where
whereafter
whereas
whereby
wherein
whereupon
wherever
whether
which
while
whither
who
whoever
whole
whom
whose
why
will
with
within
without
would
yet
you
your
yours
yourself
yourselves

Bibliography

Articles

[1] Ross Arnold and Tim Bell. A corpus for the evaluation of lossless compression algorithms. Technical report, Department of Computer Science, University of Canterbury, 1997.

[2] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:1001–1008, 1983.

[3] Adrian Barbu and Song-Chun Zhu. Graph partition by Swendsen–Wang cuts. In *Proceedings of the International Conference on Computer Vision*, 2003.

[4] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems*, volume 14, pages 577–584, 2002.

[5] Timothy Bell, I. H. Witten, and J. G. Cleary. Modeling for text compression. *ACM Computing Surveys*, 21(4):557–592, 1989.

[6] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression*. Prentice Hall, 1990.

[7] Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. A neural probabilistic language model. In *Advances in Neurological Information Processing Systems*, volume 13, pages 932–938, 2000.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[9] J. Bentley, D. Sleator, R. Tarjan, and V. Wei. A locally adaptive data compression scheme. *Communications of the ACM*, 29:320–330, 1986.

[10] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *1999 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, 1999.

[11] David Blackwell and James B. MacQueen. Ferguson distributions via Pólya urn schemes. *Annals of Statistics*, 1(2):353–355, 1973.

[12] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. In *Advances in Neurological Information Processing Systems*, volume 14, 2002.

[13] John Blitzer, Amir Globerson, and Fernando Pereira. Distributed latent variable models of lexical co-occurences. In *Proceedings of the Tenth International Workshop on Artifical Intelligence and Statistics*, 2005.

[14] Sregey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Combuter Networks and ISDN Systems*, 30:107–117, 1998.

[15] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[16] James P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the Seventeenth Annual Internation ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[17] Stanley F. Chen. *Building Probabilistic Models For Natural Language*. PhD thesis, Harvard University, 1996.

[18] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modelling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

[19] Kenneth W. Church and William A. Gale. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54, 1991.

[20] J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40:67–75, 1997.

[21] John G. Cleary and Ian H. Witten. A comparison of enumerative and adaptive codes. *IEEE Transactions on Information Theory*, 30(2):306–315, 1984.

[22] Philip J. Cowans. Information retrieval using hierarchical Dirichlet processes. In Kalervo Jarvelin, James Allan, Peter Bruza, and Mark Sanderson, editors, *SIGIR Conference on Research and Development in Information Retrieval*, volume 27, pages 564–565. ACM SIGIR, ACM Press, July 2004.

[23] Philip J. Cowans and Martin Szummer. A graphical model for simultaneous partitioning and labeling. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[24] R. T. Cox. *The Algebra Of Probable Inference*. John Hopkins University Press, 1961.

[25] A. P. Dawid. Conditional independence for statistical operations. *Annals Of Statistics*, 8:598–617, 1980.

[26] S. Deerwester, S. Dumais, T. Laundauer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. *Journal Of The American Society Of Information Science*, 41(6):391–407, 1990.

[27] Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *Annals Of Statistics*, 1(2), 1973.

[28] S. Goldwater, Thomas L. Griffiths, and M. Johnson. Interpolating between types and tokens by estimating power law generators. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

[29] T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Scioences*, 101:5228–5235, 2004.

[30] Antonio Gulli and Alessio Signorini. The indexable web is more than 11.5 billion pages. In *Proceedings of the 14th International World Wide Web conference*, 2005.

[31] Djoerd Hiemstra and Wessel Kraaij. Twenty-one at TREC-7: Ad-hoc and cross-language track. In *Text REtrieval Conference*, pages 174–185, 1998.

[32] T. Hofman. Probabilistic latent semantic indexing. In *Proceedings of the twenty-second annual SIGIR conference*, 1999.

[33] R. Nigel Horspool and Gordon V. Cormack. Constructing word-based text compression algorithms. In *Proceedings of the Data Compression Conference*, 1992.

[34] Edwin. T. Jaynes. *Probability Theory, The Logic Of Science*. Cambridge University Press, 2003.

[35] H. Jeffreys. *Theory Of Probability*. Oxford University Press, 1939. 3rd edition reprinted 1985.

[36] Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1998.

[37] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, 1980.

[38] Finn V. Jensen. *Bayesian Networks And Decision Graphs*. Springer Verlag, 2001.

[39] Marcin Kaszkiel and Justin Zobel. Effective ranking with arbitrary passages. *Journal of the American Society of Information Science*, 52(4):344–364, 2001.

[40] Slava M. Katz. Estimation of probabiliies from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, 1987.

[41] Mark D. Kerninghan, Kenneth W. Church, and William A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th Conference on Computational Linguistics.*, volume 2, pages 205–210, 1990.

[42] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal Of The ACM*, 46, 1999.

[43] Bryan Klimt and Yiming Yang. The Enron corpus: A new dataset for email classification research. In *Proceedings of the European Conference on Machine Learning*, 2004.

[44] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram langauge modeling. In *Proceedings of the IEEE International Conference on Acoutics, Speech and Signal Processing*, volume 1, pages 131–184, 1995.

[45] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algroithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[46] S Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[47] J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. In J. Lafferty W. Bruce Croft, editor, *Langauge Modeling for Information Retrieval*, pages 1–11. Kluwer Academic Publishers, 2003.

[48] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conf. on Machine Learning*, 2001.

[49] John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *2001 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, 2001.

[50] P. S. Laplace. *Philospohical Essays On probabilities*. Springer–Verlag, 1995. Originally published 1825.

[51] Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[52] G. Lidstone. Note on the general case of the Bayes–Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192, 1920.

[53] Xiaoyong Liu and W. Bruce Croft. Passage retrieval based on language models. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM)*, pages 375–382, 2002.

[54] Xiuwen Liu and DeLiang Wang. Perceptual organization based on temporal dynamics. In *Advances in Neurological Information Processing Systems*, volume 12, 2000.

[55] D. J. C. MacKay and L. Peto. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):1–19, 1994.

[56] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[57] Chris Manning and Heinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[58] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the International Conference on Computer Vision*, pages 416–425, 2001.

[59] R. J. McEliece, D. J. C. MacKay, and J. F. Cheng. Turbo decoding as and instance of Pearl's 'belief propagation' algorithm. *IEEE Journal On Selected Areas In Communication*, 16(2):140–152.

[60] Alastair Moffat. A note on the PPM data compression algorithm. Technical Report 88/7, University of Melbourne, 1988.

[61] Alistair Moffat. Word-based text compression. *Software–Practice And Experience*, 19(2):185–198, 1989.

[62] Arthur Nádas. On Turing's formula for word probabilities. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(6):1414–1416, 1985.

[63] Herman Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1994.

[64] Paul Ogilive and Jamie Callan. Experiments using the Lemur toolkit. In *Proceedings of the Text REtrieval Conference*, 2002.

[65] J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900, 1997.

[66] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *1998 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, 1998.

[67] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[68] Yuan Qi, Martin Szummer, and Thomas P. Minka. Diagram structure recognition by *b*ayesian conditional random fields. In *Proceedings of the International Conference on Computer Vision*, 2005.

[69] K. Srihari R, C. Ng, C. Baltus, and J. Kud. Use of language models in on-line recognition of handwritten sentences. In *Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition*, pages 284–294, 1993.

[70] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[71] V. V. Raghavan and S. K. M. Wong. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5):279–87, 1986.

[72] S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal Of The American Society For Information Science*, 27:129–146, 1976.

[73] S. E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[74] S. E. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[75] Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, 60(5):503–520, 2004.

[76] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992.

[77] J. Rocchio. Relevance feedback information retrieval. In Gerald Salton, editor, *The SMART retrieval system — Experiments in automatic document processing*, pages 313–323. Prentice-Hall, 1971.

[78] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. In *Computer Speech And Language*, volume 10, pages 187–228, 1996.

[79] Ronald Rosenfeld. Two decades of statistical language modelling: Where do we go from here? *Proceedings Of The IEEE*, 88(8), 2000.

[80] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A Bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, 1998.

[81] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[82] E. Saund. Finding perceptually closed paths in sketches and drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):475–491, 2003.

[83] E. Saund, D. Fleet, D. Larner, and J. Mahoney. Perceptually-supported image editing of text and graphics. In *Proc. UIST 03*, pages 183–192, 2003.

[84] Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, (4), 1994.

[85] F. Sha and F. Pereira. Parsing with conditional random fields. Technical Report MS-CIS-02-35, University of Pennsylvania, 2003.

[86] Claude E. Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, pages 50–64, 1951.

[87] P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *Uncertainty in Artificial Intelligence*, volume 4, pages 169–198, 1990.

[88] Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *International Conference on Machine Learning*, 2004.

[89] R. H. Swendsen and J. S. Wang. Nonuniversal critical dynamics in mc simulations. *Physical Review Letters*, 58(2):86–88, 1987.

[90] Martin Szummer and Philip J. Cowans. Incorporating context and user feedback in pen-based interfaces. In R. Davis and J. Landay et al., editors, *AAAI Fall symposium, Making Pen-Based Interaction Intelligent and Natural*, FS-04-06, pages 159–166. AAAI Press, 2004.

[91] Yee Whye Teh. A Bayesian interpretation of interpolated Kneser–Ney (nips workshop presentation). 2005.

[92] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. Technical Report 653, Department Of Statistics, UC Berkeley, 2003.

[93] Keith Vertanen. Efficient computer interfaces using continuous gestures, language models, and speech. Master's thesis, University of Cambridge, 2004.

[94] Hanna M. Wallach. Conditional random fields: An introduction. Technical Report MS-CIS-04-21, University of Pennsylvania, 2004.

[95] David Ward. *Adaptive Computer Interfaces*. PhD thesis, University of Cambridge, 2001.

[96] David J. Ward, Alan F. Blackwell, and David J. C. MacKay. Dasher — a data entry interface using continuous gestures and language models. In *UIST 2000: The 13th Annual ACM Symposium on User Interface Software and Technology*, 2000.

[97] David J. Ward and Davic J. C. MacKay. Fast hands-free writing by gaze direction. *Nature*, 418:838, 2002.

[98] I. H. Witten, R. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[99] Ian H. Witten and Timothy C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.

Web References

[100] A001861 from the on-line encyclopedia of integer sequences. `http://www.research.att.com/projects/OEIS?Anum=A001861`.

[101] Google book search. `http://books.google.com/`.

[102] Project Guttenberg. `http://www.gutenberg.org/`.

[103] University of Glasgow information retrieval resources. `http://ir.dcs.gla.ac.uk/resources/`.

[104] The Canterbury corpus. `http://corpus.canterbury.ac.nz`.

[105] The Dasher project. `http://www.dasher.org.uk/`.

[106] GNU Emacs website. `http://www.gnu.org/software/emacs/emacs.html`.

[107] Google. `http://www.google.com/`.

[108] Ispell spell checker. `http://www.gnu.org/software/ispell/ispell.html`.

[109] Mozilla Firefox website. `http://www.mozilla.org/products/firefox/`.

[110] MSN Search. `http://search.msn.com/`.

[111] *talks.cam* seminar database. `http://talks.cam.ac.uk/`.

[112] Tegic Communications, developers of the T9 text entry system. `http://www.tegic.com`.

[113] Text REtrieval Conferenec. `http://trec.nist.gov/`.

[114] Yahoo! `http://www.yahoo.com/`.