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,

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

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,

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

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,

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

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

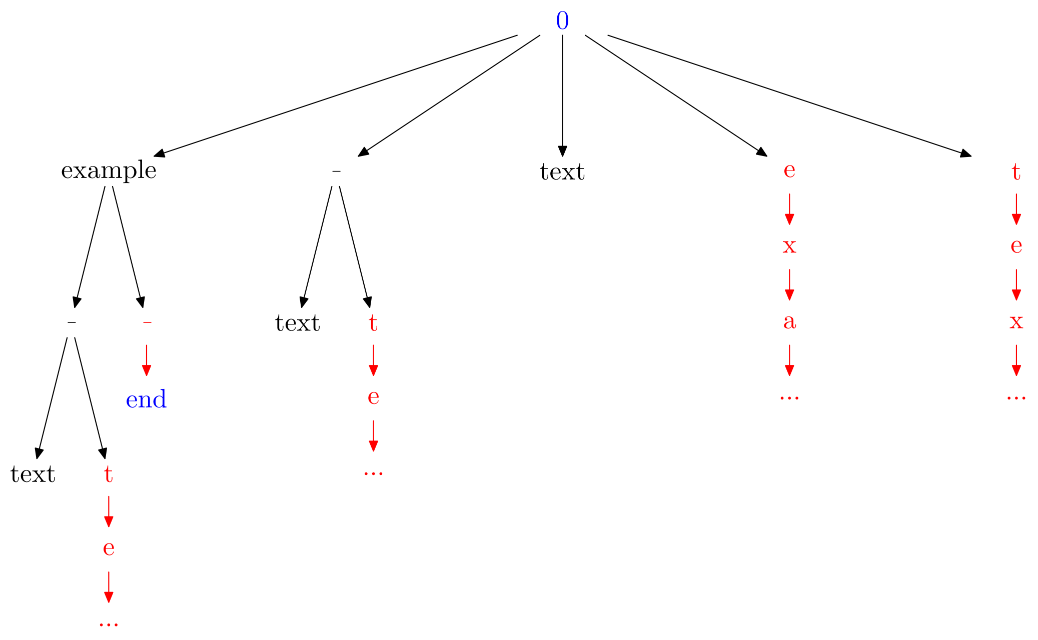

Figure 3.1: An illustration of the data structure used in the word-level model, shown storing the phrase '/example/_/text/'. Nodes representing words are shown in black, and those representing letters are shown in red. The special nodes, representing the root ('0') and the end of word marker ('end')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

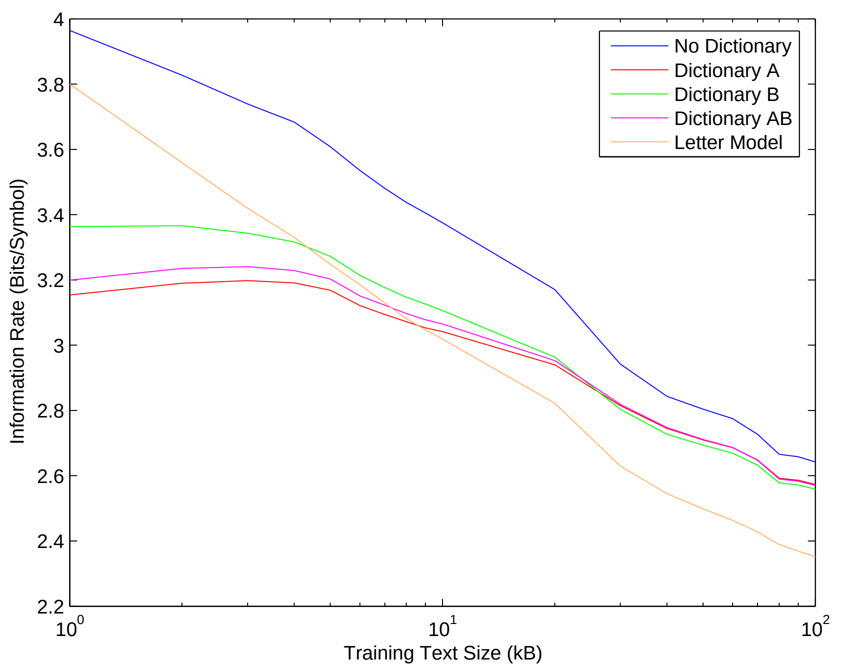

Figure 3.2: Comparison of experimental performance of the dictionary models using an in-domain test text.

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

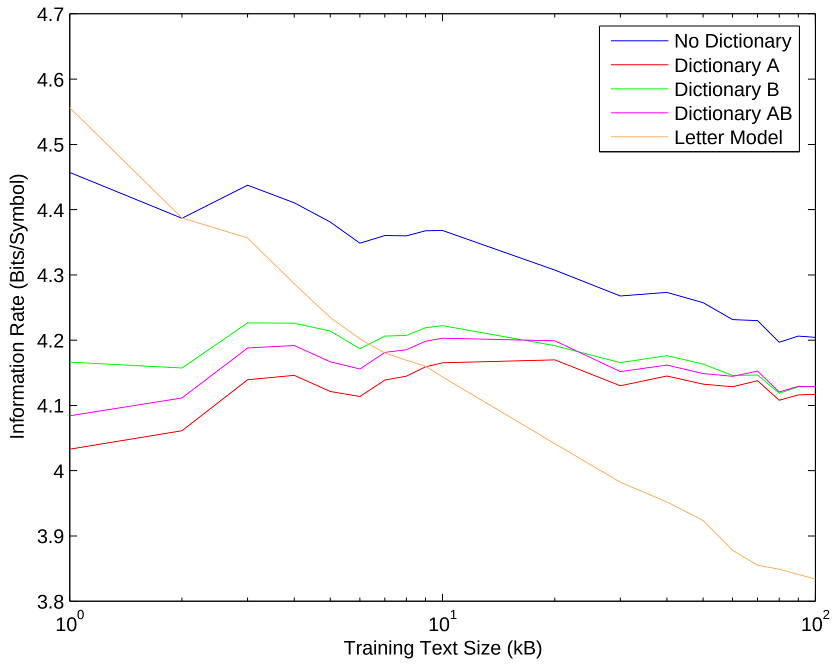

Figure 3.3: Comparison of experimental performance of the dictionary models using an out-of-domain test text.

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.

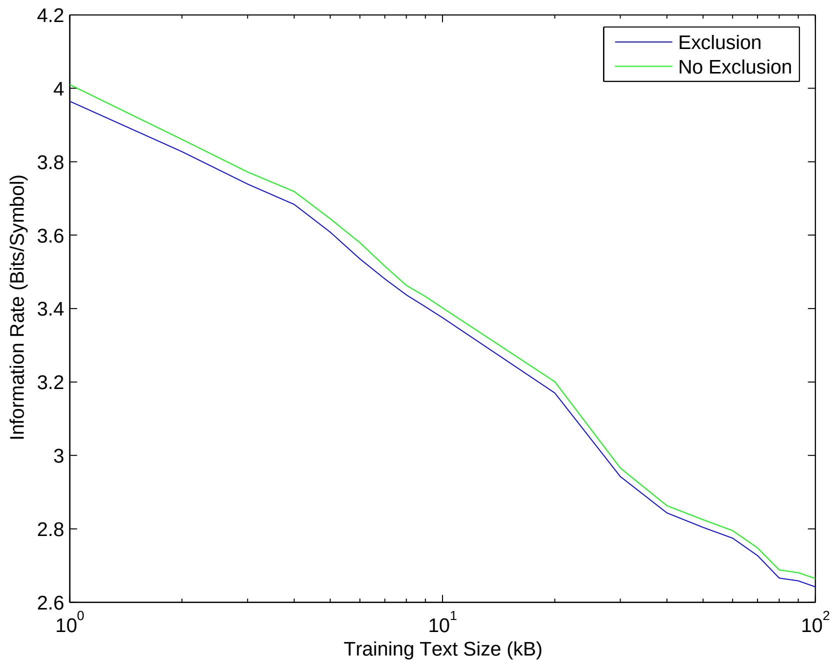

Figure 3.4: Effect of update exclusion when entering the symbol-level model.