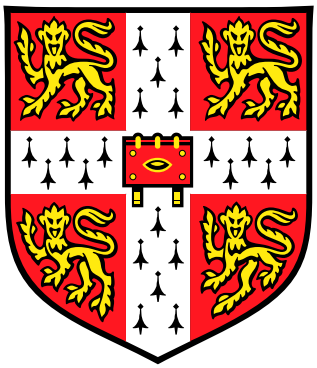
Log-Linear Interpolation of Language Models

Alexander Gutkin
Peterhouse



University of Cambridge

19th November 2000

Thesis submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of
Master of Philosophy
in
Computer Speech and Language Processing

Abstract

Building probabilistic models of language is a central task in natural language and speech processing allowing to integrate the syntactic and/or semantic (and recently pragmatic) constraints of the language into the systems. Probabilistic language models are an attractive alternative to the more traditional rule-based systems, such as context free grammars, because of the recent availability of massive amount of text corpora which can be used to efficiently train the models and because instead of binary grammaticality judgement offered by the rule-based systems, likelihood of any sequence of lexical units can be obtained, which is a crucial factor in such tasks as speech recognition. Probabilistic language models also find their application in part-of-speech tagging, machine translation, semantic disambiguation and numerous other fields.

The most widely used language models are based on the estimation of the probability of observing a given lexical unit conditioned on the observations of $n-1$ preceding lexical units, and are known as n-gram models. When the n-gram estimates are poor, whatever the reason for that may be, a technique called smoothing is applied to adjust the estimates and hopefully produce more accurate model. Smoothing techniques may be roughly divided into the backing-off and interpolation. In the first case, the best n-gram model in the current context is selected, whereas in the second case all the n-gram models of different specificities are combined together to form a better predictor.

In this thesis, a recently proposed novel interpolation scheme is investigated, namely, the log-linear interpolation. Unlike the original publication, however, which dealt with combining the models of unrelated nature, the aim of this thesis is to formulate the theoretical framework for smoothing the n-gram probability estimates obtained from similar language models with different levels of specificity on the same corpus, which will be called log-linear n-gram smoothing, and compare it to the well-established linear interpolation and back-off methods. The framework being proposed includes probability combination, parameter optimisation, dealing with data sparsity and parameter clustering.

The resulting technique is shown to outperform the conventional linear interpolation and back-off techniques when applied to the n-gram smoothing tasks.

Acknowledgements

First and foremost I would like to thank my supervisor, Dr Thomas Niesler who has encouraged, supported and guided my language modelling endeavours during every step of the way and without whose sound and friendly advice completion of this dissertation would not be possible. My deepest professional gratitude goes to him.

I owe gratitude to William Lee of Department of Earth Sciences for proofreading this thesis and making several insightful comments on mathematical aspects of presentation without which this thesis would not appear as it appears now, and to Valerie Dorrzapf of Department of English and Applied Linguistics, who supplied interesting insights on the things I have been doing from the theoretical linguistics perspective.

Completion of the experiments described herein has been made possible by Patrick Gosling who continues to maintain excellent facilities for the Speech Group here at Fallside Laboratory.

I would also like to thank people who first introduced me to language modelling in speech and natural language processing, namely Dr Ted Briscoe for his excellent course of lectures on stochastic context free and unification based grammars and Phil Woodland for his expert professional coverage of language modelling for speech recognition, as well as following people, postgraduates and members of staff alike, Dr. Mark Gales, Aurélien Max, Paolo Mendonça, Dan Povey, Nathan Smith, and Matt Stuttle, whose friendly attitude and high expertise proved to be a deeply rewarding experience for me. I want to thank Dr Stephen Pulman for explaining to me the intricate differences between mathematical and computational linguistics and pointing at the inherent difficulties in the existing approaches.

Finally, it would be unfair not to mention Simon Redhead and Kathrin Schödel and thank them for all those numerous coffee/tobacco breaks and laughs on the balcony regardless of the time and weather conditions, whose excellent company has helped me to enjoy my life in Cambridge, and thank Pál Hegedüs and Réka Limbek for everything.

Declaration

This dissertation is the result of my own original work, and where it draws on the work of others, this is acknowledged at the appropriate points in the text. This dissertation has not been submitted in whole or in part for a degree at any other institution. The length of this thesis, including footnotes, is approximately 14,815 words. The implementation of the theoretical frameworks and algorithms derived in this work can be found in appendix to this dissertation which is submitted separately.

To my family.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Language modelling is the attempt to characterise, capture and exploit syntactic, semantic and pragmatic regularities exhibited by natural language. It is being widely used in many domains including speech recognition, optical character recognition, handwriting recognition, machine translation, part-of-speech tagging, dialog modelling and spelling correction. In its simplest form, a language model may be a representation of the list of the sentences belonging to a language, while the more complex models may also try to describe the structure and meaning underlying the sentences in a natural language.

Techniques for modelling the language historically fall into two categories. The first type of models are the traditional grammars, such as context free and unification based grammars, which although being rigorously defined from linguistic perspective, suffer from the typical deficiencies of the rule-based systems. They are difficult to maintain, adapt to the new domains and languages, and their computation complexity is too high to be efficiently employed in time critical applications, such as large vocabulary continuous speech recognition. Although linguistically appealing, these models are out of scope in this thesis. In more recent years, the second language model category, namely the corpus-based shallow probabilistic models, based on statistical representation of the natural language, have gained common usage.

A statistical language model describes probabilistically the constraints on word order found in language: typical word sequences are assigned high probabilities, while atypical ones are assigned low probabilities. Statistical models of language may be evaluated by measuring the predicted probability of unseen test utterances: models that generate high average word probability (equivalent to low novelty or low entropy or low perplexity) are considered superior. The perplexity measure is commonly used as a measure of "goodness" of such a model. The most widely used statistical model of language is the n-gram model, in which an estimate of the likelihood of a word w_n is made solely on the identity of the preceding $n-1$ words in the utterance. The strengths of the n-gram model come from its success at capturing local constraints, the ease by which it may be constructed from text corpora, and from its computational efficiency in use.

One of the problems exhibited by the statistical prediction of the natural language is the problem of *data sparseness* arising the uneven distribution of lexical units in language. For n-gram models, for instance, most of the possible n-tuple events are never encountered in the text corpus used to train the model, regardless of the size of the corpus and in order for such a language model to be reliable, it must be ensured that the probabilities that this model assigns to the word strings

are nonzero, otherwise the "unseen" word sequences in question will be rendered improbable and will not be hypothesised. A technique used in language modelling for obtaining accurate probability estimates when there is insufficient amount of training data is called *smoothing*. Smoothing overcomes the shortcomings of the conventional probability estimates by taking into the account the following considerations

- All word combinations are possible, *i.e.* there is not a single word sequence with zero probability.
- Because of the nonlinear distribution of words in natural language, the amount of training data can always be assumed to be small, even if its size is in millions of words, *i.e.* there are always going to be events in the evaluation corpora which are unobserved in the training corpus.

The simplest smoothing techniques achieve this effect by pretending that each n-gram occurs once more than it actually did, while the more complicated ones define complex discounting frameworks.

In this thesis, the discussion is restricted to the smoothing of an n-gram models, where the structure of the model is unchanged but where the method used to estimate the probabilities of the model is modified. There are numerous other types of the language models to which smoothing can be applied, such as class-based language models [9], maximum entropy models [64], decision tree based models [1] and stochastic grammars [69] [8] and it remains to be seen whether improved smoothing techniques for the n-gram language modelling can lead to improved performance for these other models.

Language model smoothing frameworks usually fall into two basic categories, the first based on selecting the best model in the current context among the available ones as a predictor, with probably the best known model being the back-off model suggested by Katz [37], while the second ones usually combine all of the language models together to obtain a probability estimate, with linear interpolation used by Jelinek [32] being the typical representative.

This thesis is concerned with the special case of smoothing belonging to the second basic category, namely combination of all the language models. In this work, a novel interpolation technique called *log-linear interpolation* is investigated.

1.1 Scope of the Thesis

The problems that have been selected investigate the use of log-linear interpolation for n-gram language model smoothing at following different levels: framework for accurate probability estimation, efficient parameter optimisation and parameter tying.

Accurate probability estimation framework is important because the task of a language model is to supply reliable estimates, even under the exceptional conditions, such as manifestations of data sparsity problem. In particular, when there is not enough data available, log-linear interpolation attempts to remedy this by using the combination of lower-order models. Several additional issues addressed concern choice and calculation of the normalisation factors for the model.

The parameters of a log-linear interpolation model should be optimal with respect to the training data, and yield satisfactory performance on the test corpora.

Different optimisation algorithms are proposed which make such an efficient parameter estimation possible.

Finally, in order for the log-linear interpolation parameters controlling the performance of the model to be estimated reliably, there should be enough training data made available to the model. If the amount of the training data is not sufficient, the problem is solved by constraining certain groups of the parameters to have the same value, *i.e.* tying them. In this context, possible parameter tying algorithm for log-linear interpolation is proposed.

Because of the widespread use of the aforementioned linear interpolation and back-off language models, they were selected as the baseline models with which the theoretical and experimental results obtained for the log-linear interpolation are compared.

1.2 Thesis Organisation

This dissertation is organised as follows: Chapter 2 provides a necessary background to language modelling, chapter 3 discusses the conventional smoothing techniques prevalent in language modelling, chapter 4 provides the theoretical framework for linear and log-linear interpolation in the context of smoothing, namely the techniques for parameter clustering, optimisation and efficient probability estimation and chapter 5 describes the experiments carried out with the interpolation models developed in this dissertation and presents some interesting results obtained for the novel log-linear smoothing framework for language modelling. Finally, chapter 6 presents summary and conclusion.

CHAPTER 2

Background to Language Modelling

The task of *language modelling* is to assign a probability value to every possible word in a text stream based on its likelihood of occurrence in the context in which it finds itself. This task is fundamental to speech and optical character recognition, as well as to many areas of natural language processing, such as word sense disambiguation and probabilistic parsing.

In speech recognition there is need to calculate probabilities $P(\boldsymbol{w})$ of word strings $\boldsymbol{w} = w_1, \ldots, w_n$, where each word w_i belongs to a fixed and known vocabulary. Using the definition of conditional probabilities following decomposition can be obtained

$$P(\boldsymbol{w}) = \prod_{i=1}^{n} P(w_i | w_1, \ldots, w_{i-1}) \ , \tag{2.1}$$

where $P(w_i|w_1, \ldots, w_{i-1})$ is the probability of a word w_i being spoken given that words $w_1, \ldots, w_{i-1}$ were uttered previously. The task of a statistical language model, therefore, is to provide the decoder with adequate estimates of the probabilities $P(w_i|w_1, \ldots, w_{i-1})$.

The word string $w_1, \ldots, w_{i-1}$ in (2.1) is usually referred to as *history* of the word w_i. It should be noted that for a vocabulary of size N there are N^{i-1} possible distinct histories and N^i values are needed for complete specification of probabilities $P(w_i|w_1, \ldots, w_{i-1})$. For practical vocabulary sizes such an astronomical number of estimates can neither be stored nor accessed efficiently.

2.1 Equivalence Classification of History

In order to avoid the problem mentioned above, all possible conditioning histories $w_1, \ldots, w_{i-1}$ must be distinguished as belonging to some manageable number N_H of equivalence classes [32]. It is therefore desirable to define a many-to-one (in some applications many-to-many [9] [29] [59]) mapping operator $H(\cdot)$ that would classify a given history $w_1, \ldots, w_{i-1}$ of word w_i as belonging to one of N_H subsets h_k, *i.e.*

$$H(w_1, \ldots, w_{i-1}) = h_k \quad k \in [0, N_H - 1] \ ,$$

where the set of equivalence classes $\mathcal{H}$ is given by

$$\mathcal{H} = \{h_0, \ldots, h_{N_H - 1}\} \ .$$

For many-to-one mapping, conditional probabilities given in (2.1) may now be estimated as

$$P(\boldsymbol{w}) = \prod_{i=1}^{n} P(w_i|H(w_1,\ldots,w_{i-1})) \ . \tag{2.2}$$

The problem therefore is to define an appropriate mapping operator to be used in (2.2). The most popular approach is to assume that the dependence of the conditional probability of observing a word w_i at position i is restricted to its prior local context, *i.e.* to its immediate n predecessor words $w_{i-n},\ldots,w_{i-1}$. This is essentially a Markov chain assumption which leads directly to notion of n-gram language models for which

$$H(w_1,\ldots,w_{i-1}) \triangleq w_{i-n+1},\ldots,w_{i-1} \ . \tag{2.3}$$

The most widely used n-gram models are obtained for $n = 1$ (*bigram*) and $n = 2$ (*trigram*).

Number of alternative equivalence classifiers, which lie outside the scope of this discussion, have been developed over the past decade, *e.g.* application of decision trees to clustering of the word histories [1] [6] [24].

2.2 Statistical Estimation

Given a training corpus of size N representing some language of interest and history equivalence classification that divides the training corpus into N_H subsets, the second goal is to find out a way to derive a reliable probability estimates for the words in the corpus given their histories. The following sections describe various specialised statistical techniques to obtain such estimates. Before commencing, several notions should be defined. Throughout the chapter, *counts* will be used to describe the training data $w_1,\ldots,w_i,\ldots,w_N$. As an example, trigram counts $N(u,v,w)$ are obtained by counting how often the particular word trigram (u,v,w) occurs in the training data[1]

$$N(u,v,w) = \sum_{i:(w_{i-2},w_{i-1},w_i)=(uvw)} 1 \ .$$

Following count definitions are used:

$N(h,w)$	number of observations for joint event (h,w);
$N(w)$	number of observations for word w;
$N(h)$	number of observations for history h;
N	total number of observations.

In addition, *count-counts* or *frequencies of frequencies* n_r and $n_r(h)$ are defined as how often a certain count r has occurred, *i.e.*

$n_r(h)$	number of distinct words w that were seen following history h exactly r times;
n_r	total number of distinct joint events (h,w) that occurred exactly r times.

For $r = 0$ the events are called unseen (never observed in the training data) and for $r = 1$ the events are called singleton events (observed exactly once). As we shall see later n_0 and n_1 play crucial role in estimation from sparse data.

[1] Sometimes counts are referred to as *relative frequencies*.

2.2.1 Maximum Likelihood estimation

For each word w_i in a position i of a text corpus $w_1, \ldots, w_i, \ldots, w_N$ its conditioning history h_i is known. To arrive at maximum likelihood estimate for the set of conditional probabilities $\{P(w|h)\}$ consider the logarithm of the likelihood $G(\{P(w|h)\})$ which has to be optimised over the set $\{P(w|h)\}$ [58]:

$$\begin{aligned} G(\{P(w|h)\}) &= \log \prod_{i=1}^{N} P(w_i|h_i) \\ &= \sum_{i=1}^{N} \log P(w_i|h_i) \\ &= \sum_{h,w} N(h,w) \log P(w|h) \ , \end{aligned} \tag{2.4}$$

where in the last line of (2.4) the summation index has been changed by using the count definitions $N(h,w)$. In addition, following normalisation constraint must be observed while optimising log-likelihood function

$$\sum_{w} P(w|h) = 1, \quad \forall h \in \mathcal{H} \ . \tag{2.5}$$

Given (2.4) and (2.5), following function which includes the constraints is optimised using the method of Lagrangian multipliers

$$\tilde{G}(\{P(w|h); \mu_h\}) = \sum_{h,w} N(h,w) \log P(w|h) - \sum_{h} \mu_h \left[\sum_{w} P(w|h) - 1 \right] \ . \tag{2.6}$$

By taking partial derivatives with respect to each of the probabilities $P(w|h)$ in (2.6) and each of the Lagrangian multipliers μ_h and equating them to zero we obtain following set of equations

$$\begin{aligned} \frac{\partial \tilde{G}}{\partial P(w|h)} &= \frac{N(h,w)}{P(w|h)} - \mu_h \quad = 0 \ , \\ \frac{\partial \tilde{G}}{\partial \mu_h} &= \sum_{w} P(w|h) - 1 \quad = 0 \ . \end{aligned} \tag{2.7}$$

As can be seen, the second equation in (2.7) expresses exactly the normalisation constraint for each history h. By some straightforward manipulations maximum likelihood estimate for word w given its history h thus becomes

$$P_{\mathrm{ML}}(w|h) = \frac{N(h,w)}{\sum_{w'} N(h,w')} = \frac{N(h,w)}{N(h)} \ . \tag{2.8}$$

It can be seen that such an estimate assigns the highest probability to the training corpus and does not waste any probability mass on events not observed during the training. Therefore it follows that estimator of the form (2.8) will assign zero probability to any event not seen in the training corpus. The problem of unseen events is linked directly to the notion of data sparseness.

2.2.2 Sparse data problem

The fundamental problem of language modelling is the problem of *data sparseness*. While a few words in a language of interest are common, the vast majority of words are very uncommon - and longer n-grams involving them are thus much rarer again. Such n-grams will be assigned zero probabilities by the maximum likelihood estimator if such n-grams were not seen during the training. These zero probabilities will then be propagated in (2.2) which will result in wrong estimates for sentences. Experiments with training corpus of 1.5 million words described in [3] have shown that 23% of the trigram tokens found in further test corpus (which only contained 300.000 words) were previously unseen.

Assuming that the size of the corpus is not big enough one might hope that by collecting more data it would be possible to avoid the problem of data sparseness. While this may initially seem plausible (by increasing the coverage of the training corpus it is possible to refine the existing probability estimates and obtain additional ones) there is no general solution to the problem. While there is a limited number of frequent events in the language under investigation, there is a seemingly never ending tail to the probability distribution of rarer and rarer events and by simply collecting more and more data one would never reach the end of the tail.

This phenomenon was first observed by Zipf [74] [75] who uncovered the following pattern of statistical distribution of language: By obtaining the counts for each word type in a corpus and sorting the word types in order of their frequency of occurrence, the relationship between counts (frequency) for a word $N(w_i)$ and its position in the list (known as *rank* order of frequency) $r(w_i)$ is roughly a reciprocal curve, *i.e.*

$$N(w_i) \propto \frac{1}{r(w_i)}$$

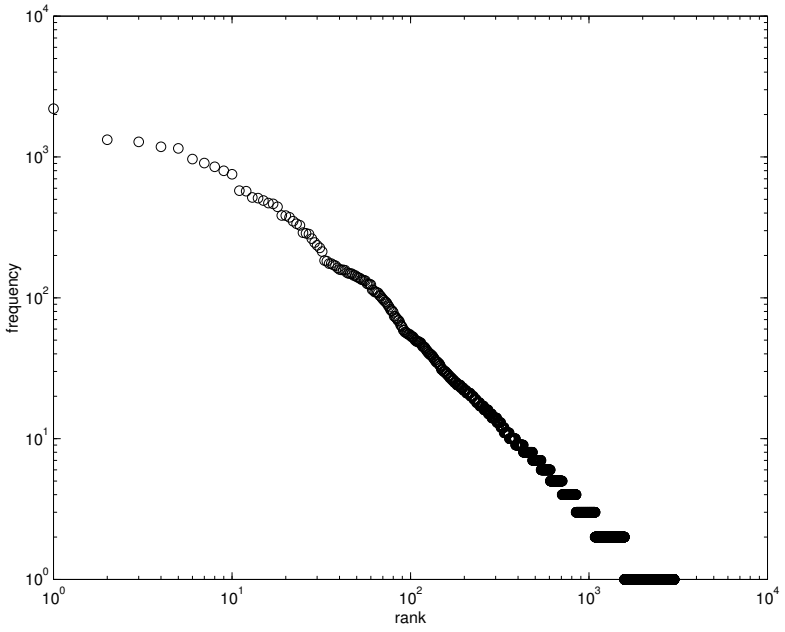
or equivalently, there exists a constant k such that for all different word types w_i in the corpus,

$$N(w_i)r(w_i) = k \ . \tag{2.9}$$

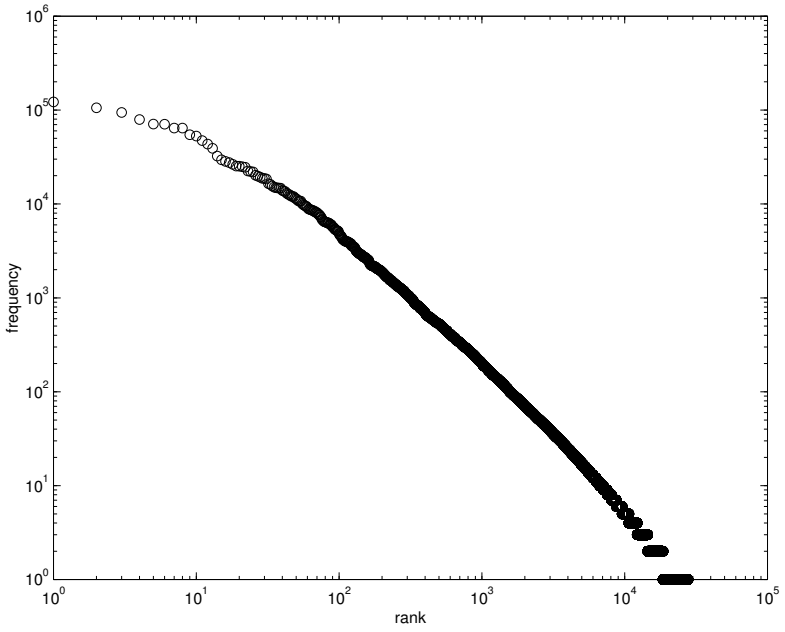
While (2.9) is only a rough approximation (for an attempt to find a closer fit to the empirical distribution of words consult Mandelbrot [48] [49] and Sichel [67]), it is still useful as a description of the frequency distribution of words in human languages: there are few very common words, a middling number of medium frequency words and many low frequency words. Figure 2.1 shows rank-frequency plots of the words for two different corpora where 2.1(a) corresponds to a small evaluation corpus consisting of approximately $47{\times}10^3$ word tokens and 3000 word types and 2.1(b) corresponds to a bigger training corpus consisting of $3.6{\times}10^6$ word tokens and $28{\times}10^3$ word types. These specific text corpora are described in detail in section 5.1.

As can be seen, even after the significant increase in size of the corpus, same form of statistical distribution underlying the language is obtained, rendering approaches based on maximum likelihood impractical. Therefore there is need to devise better estimators that allow for possibility of observing the yet unseen events and making all word combinations possible (*i.e.* by assigning non-zero probabilities) even if the amount of training data is insufficient.

There is still an ongoing debate about the nature of the processes described by Zipf's law, best summarised by a renowned linguist George Miller, who wrote in 1965 [75]:



(a) `Hub5-Eval98` evaluation corpus



(b) `Hub5-Train00` training corpus

Figure 2.1: Rank-frequency plot of the words (unigrams) on doubly logarithmic axes for small evaluation and big training corpora showing the same pattern of underlying Zipfian distribution.

> Faced with this massive statistical regularity, you have two alternatives. Either you can assume that it reflects some universal property of human mind, or you can assume that it represents some necessary consequence of the laws of probabilities. Zipf chose the synthetic hypothesis and searched for a principle of least effort that would explain the apparent equilibrium between uniformity and diversity of our use of words. Many others who were subsequently attracted to the problems chose the analytic hypothesis and searched for a probabilistic explanation...

Some of the recent findings supporting the synthetic hypothesis can be found in [70], while the arguments supporting the analytic hypothesis are presented in [45], [46] and [36]. Paper by Silagadze [68] contains an interesting overview of the research efforts in other, non-linguistic areas where Zipf's law still applies.

2.3 Quality Assessment of Language Models

The most common evaluation techniques for language modelling are based on information theory. Assuming that a language is an information source emitting a sequence of symbols $\boldsymbol{w}$ from a finite vocabulary $\mathcal{V}$ (*i.e.* viewing it as a stochastic process), it is characterised by its inherent entropy H which is defined as the amount of non-redundant information conveyed per word, on average, by a certain language L in question. The *entropy* $H(X)$ of a discrete random variable X is defined as

$$H(X) = -\sum_{x \in X} P(x) \log P(x) = -E_P \left(\log P(X) \right) \ , \tag{2.10}$$

where the log is to the base of 2 expressing the entropy in bits and E_P is an expectation of a random variable $\log P(X)$.

For the two probability mass functions $P(X)$ and $Q(X)$ their *relative entropy* or *Kullback-Leibler distance* is given by

$$D(P\|Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)} = E_P \left(\log \frac{P(X)}{Q(X)} \right) \ . \tag{2.11}$$

The quantity in (2.11) is always non-negative and equals to zero if and only if the two probability distributions $P(X)$ and $Q(X)$ are identical [19].

The *cross-entropy* between a random variable X with true probability distribution $P(X)$ and another probability distribution $Q(X)$ is defined as

$$H(X, Q) = H(X) + D(P\|Q) = -\sum_{x \in X} P(x) \log Q(x) \ .$$

It is therefore possible to introduce the cross-entropy of a language L described by $P_L(X)$ with respect to a certain model M as

$$H(P_L, P_M) = -\sum_{x \in X} P_L(x) \log P_M(x) \ . \tag{2.12}$$

According to Shannon-McMillan-Breiman theorem[2] [19] the entropy of equation (2.10) can be defined as

$$H(X) = -\lim_{n \to \infty} \frac{1}{n} \log P(X_1, X_2, \ldots, X_n) \ ,$$

[2] Also known as Asymptotic Equipartition Property.

which allows to rewrite the cross-entropy in (2.12) as

$$\begin{aligned} H(P_L, P_M) &= -\lim_{n\to\infty} \frac{1}{n} \sum_{x_1,x_2,\ldots,x_n} P_L(x_1, x_2, \ldots, x_n) \log P_M(x_1, x_2, \ldots, x_n) \\ &= -\lim_{n\to\infty} \frac{1}{n} E_{P_L} \left(\log P_M(x_1, x_2, \ldots, x_n)\right) \ . \end{aligned}$$

Assuming that P_L is ergodic, this simplifies to

$$H(P_L, P_M) = -\lim_{n\to\infty} \frac{1}{n} \log P_M(X_1, X_2, \ldots, X_n) \ ,$$

which for a "sufficiently" large sample size n can be simplified as

$$H(P_L, P_M) \approx -\frac{1}{n} \log P_M(X_1, X_2, \ldots, X_n) \ . \tag{2.13}$$

The above quantity is very useful since it specifies an upper bound on the unknown true entropy $H(P_L)$ of the language, *i.e.* for any model M,

$$H(P_L) \leq H(P_L, P_M) \ ,$$

where the difference between $H(P_L, P_M)$ and $H(P_L)$ is a measure of inaccuracy of the model M with respect to the true model L.

Cross-entropy estimate of (2.13) is the most commonly used metric for evaluating the performance of language models. Given the test corpus $\mathcal{T}$, composed of $n_{\mathcal{T}}$ sentences $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n_{\mathcal{T}}}$, which is disjoint from the data used to train the model M, equation (2.2) is used to calculate the probability of a sentence $\boldsymbol{w}_i$ given the model allowing to calculate the cross-entropy of the test corpus $\mathcal{T}$ using the following

$$H(P_{\mathcal{T}}, P_M) = -\frac{1}{S_{\mathcal{T}}} \log \prod_{i=1}^{n_{\mathcal{T}}} P(\boldsymbol{w}_i) \ , \tag{2.14}$$

where $S_{\mathcal{T}}$ is the total number of word tokens in $\mathcal{T}$.

By using simple trigram models, smoothed using the linear interpolation, trained on slightly more than half a billion words drawn from various corpora assumed to be reasonable representative sample of English and tested on Brown corpus [41] [21] of one million words, Brown *et al.* [10] give an upper bound of 1.75 bits per character for the entropy of English, which is actually higher than the original Shannon's [66] estimate of 1.3 bits per character, who used human subjects for obtaining this gambling estimate.

An alternative metric, directly related to cross-entropy, is *perplexity* [32] defined as a reciprocal of the geometric average probability assigned by the model M to each word in the test corpus $\mathcal{T}$, *i.e.*

$$PP_M(\mathcal{T}) = 2^{H(P_{\mathcal{T}}, P_M)} \ ,$$

where the log is assumed to have the base of 2, which need not necessarily be, however[3]. Given the probability estimate of a sentence of size n defined in (2.1) and using the cross-entropy estimate from (2.13), the perplexity can now be expressed as

$$\begin{aligned} PP_M(\mathcal{T}) &= 2^{-\frac{1}{n} \log \prod_{i=1}^{n} P_M(w_i|w_1,\ldots,w_{i-1})} \\ &= 2^{-\frac{1}{n} \sum_{i=1}^{n} \log P_M(w_i|w_1,\ldots,w_{i-1})} \\ &= P_M(w_1, \ldots, w_n)^{-\frac{1}{n}} \ , \end{aligned}$$

[3] In consequent experiments log is taken to have the base of 10.

which can be thought of as measure of complexity of a task of recognising a text with $PP_M(\mathcal{T})$ equally likely words presented to a language model.

Minimising the perplexity is analogous to minimising the cross-entropy of the model with respect to the test set. The goal of statistical language modelling therefore can be viewed as minimising the perplexity (cross-entropy) so as to bring it down as close as possible to the true entropy of the language.

CHAPTER 3

Overview of Smoothing Techniques

This chapter described some of the most popular techniques used in language modelling for obtaining more reliable probability estimates by applying the technique called *smoothing*. Section 3.1 presents various discounting techniques used as a basis for forming the statistical estimators and section 3.2 presents the models comprised of the combination of statistical language model estimators using various methods of discounting which usually yield more reliable and robust predictors.

3.1 Discounting Methods

This section presents an overview of the discounting methods prevalent in language modelling which try to remedy the data sparseness problem manifested by the natural languages.

3.1.1 Basic discounting

An alternative form of maximum likelihood estimate in (2.8) for a given n-gram (h, w) is

$$P_{\mathrm{ML}}(h, w) = \frac{N(h, w)}{N} \quad . \tag{3.1}$$

Therefore the probability estimate for an n-gram seen r times becomes

$$P_{\mathrm{ML}}(N(h, w) = r) = \frac{r}{N} \quad . \tag{3.2}$$

The basic idea behind the following approaches is to remove some probability mass from the observed events and assign it for events which were unseen during the training. The oldest solution is to employ *Laplace's law of succession* (sometimes referred to as *adding one*) [43] [44] which adds one (phantom) observation to every frequency count required to obtain MLE for a given model. Then (3.1) becomes

$$P_{\mathrm{Lap}}(h, w) = \frac{N(h, w) + 1}{N + S} \quad , \tag{3.3}$$

where S is the set of all distinct n-grams being considered (*i.e.* the number of phantom observations). Expressed in notation of (3.2) the modified frequency count may be written as

$$r^* = P_{\mathrm{Lap}}(r)N = \frac{(r + 1)N}{N + S} \quad . \tag{3.4}$$

Since $S \gg N$, given the Zipfian distributions with long tails of infrequent events, (3.4) tends to assign too much of the probability mass to the unseen events.

A variant of Laplace's law defined in (3.3) usually involves adding some positive value of δ smaller than one. This technique is known as Lidstone's law [30] [44] [63] for which

$$P_{\text{Lid}}(h, w) = \frac{N(h, w) + \delta}{N + \delta S} \tag{3.5}$$

and

$$r^* = \frac{(r + \delta)N}{N + \delta S} \ . \tag{3.6}$$

While (3.3) and (3.4) have effect of assuming a uniform uninformative prior on events and applying a Bayes estimator, (3.5) and (3.6) can be viewed as linear interpolation between an MLE estimate and a uniform prior [7].

Although Lidstone's law seems to help to avoid the problem, manifested by Laplace's approach, of taking too much probability mass off the observed events by choosing a small value of δ, two major objections prove its ineffectiveness: (i) there should be some way of guessing an appropriate value of δ in advance, and (ii) such simple discounting schemes yield probability estimates linear in MLE frequency, which is not a good match to the empirical distribution at low frequency [49]. Overall both techniques mentioned above have been shown to perform poorly [22].

3.1.2 Good–Turing estimate

The Good–Turing estimator is central to many smoothing techniques. The initial development was the derivation of this important formula for the field of biology where it has been widely used [27]. The result can be stated as a theorem which is presented below (both formulation of the theorem and its rigorous proof may be found in [17] and [23]) to place the subsequent developments in the proper context. At the end of this section some implications of this important result are discussed.

Let s_v, $v = 1, \ldots, S$ be a finite collection of types (words, bigrams or species of animals). For each type, tokens (examples of words or bigrams, etc.) can be sampled[1]. Let $B(N; p_1, \ldots, p_S)$ denote a sample of size N [2] drawn from S types, each type s_v having a binomial distribution with probability P_v. Let n_r be the number of types frequency of which in a sample is r and let r_v denote the frequency of the vth type.

> Good's Theorem: *When two independent marginally binomial samples $B_1(N; p_1, \ldots, p_s)$ and $B_2(N; p_1, \ldots, p_s)$ are drawn, the expected frequency r^* in the sample B_2 of types occurring r times in B_1 is*
>
> $$r^* = \frac{(r + 1)}{(1 + 1/N)} \frac{E(n_{r+1}|B(N + 1; p_1, \ldots, p_s))}{E(n_r|B(N; p_1, \ldots, p_s))} \ .$$

For a practical sizes of samples N, it immediately follows that

$$r^* \approx (r + 1) \frac{E(n_{r+1}|B(N))}{E(n_r|B(N))} \ . \tag{3.7}$$

[1] For a unigram case, S would essentially be equal to the size of the vocabulary $|\mathcal{V}|$, for a bigram case, $S = |\mathcal{V}|^2$, etc.

[2] *e.g.* a text corpus

This assumption introduces a relative error of $1/N$. For a practical computation, the expectations in (3.7) are estimated by smoothed values $S(\cdot)$ yielding

$$r^* \approx (r+1)\frac{S(n_{r+1})}{S(n_r)} \quad . \tag{3.8}$$

For the unsmoothed count-counts, (3.8) takes the form of a Turing estimator [54]

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \quad , \tag{3.9}$$

whereas (3.8) is referred to as Good–Turing estimator, which may alternatively be derived using cross-validation approaches described in [33] and [58] discussed later in this chapter.

Substitution of the empirical estimates n_r for the expectations $E(n_r)$ cannot be done uniformly since n_r will be unreliable for the high values of r. In particular, the most frequent type would be estimated by (3.9) to have probability zero, since the number of types with frequency one greater than it is zero. This prevents (3.9) from being used directly.

The original solution proposed in [27] is to fit some function $S(\cdot)$ through the observed values of (r, n_r) and to use the smoothed values for the expectations, leading to (3.8). Many different Good–Turing estimators are possible depending on how the smoothing is performed. Note that the calculation for $r = 0$ rests on knowing n_0, the number of types not observed during the training, which can be calculated given the vocabulary size $|\mathcal{V}|$. As an example, consider bigrams for which the total universe of types to estimate is $S = |\mathcal{V}|^2$. Therefore

$$n_0 = |\mathcal{V}|^2 - \sum_{r>0} n_r \approx |\mathcal{V}|^2, \quad \sum_{r>0} n_r < N \ll |\mathcal{V}|^2 \quad , \tag{3.10}$$

where for practical size of the vocabulary, approximation in (3.10) is a safe assumption.

An interesting theoretical and empirical comparison between Good–Turing estimator and Zipf's Law can be found in [65].

3.1.3 Cross-validation (deleted estimation)

The result presented in Sect.3.1.2 is derived under the assumption that the distribution of each type is binomial, which essentially means that events occur independently of each other [30][3].

An empirical realisation of Good's result is the *held-out estimator* [35] [33] for which the available training corpus is divided into retained and held-out parts (the general name given to methods using held-out and retained sets is *cross-validation* [20]). The assumption behind this and the following methods is weaker than Good's binomial assumption and simply states that both parts of the text are generated by the same process [17]. The basic held-out estimation is done as follows: Denoting by $N_1(\cdot)$ and $N_2(\cdot)$ the counts for the retained and held-out sets respectively and letting n_r denote the number of n-grams with frequency r in the retained set, all occurrences of all the n-grams with frequency r in the held-out set are counted as

$$c_r \triangleq \sum_{\{hw: N_1(h,w)=r\}} N_2(h,w)$$

[3] This is not necessarily a good assumption.

and the adjusted frequency r^* is calculated as

$$r^* = \frac{c_r}{n_r} \ . \tag{3.11}$$

Rather than using some of the training data only for frequency counts and some only for smoothing probability estimates, more efficient schemes are possible where each part of the training data is used both as initial training (retained) data and as held-out data. A method which makes more efficient use of training data than the basic held-out estimation is *deleted estimation* [35] [33]. Denoting the two parts of the training data by 0 and 1, n_r^0 is the number of types occurring r times in the 0th part and c_r^{01} is the total number of occurrences of these types in the 1st part. Likewise, n_r^1 is defined as number of types occurring r times in the 1st part of training data and c_r^{10} number of occurrences of these types in the 0th part. The two basic held-out estimates are c_r^{01}/n_r^0 and c_r^{10}/n_r^1 which are then combined to yield an equation for deleted estimator

$$r^* = \frac{c_r^{01} + c_r^{10}}{n_r^0 + n_r^1} \ . \tag{3.12}$$

Experiments described in [17] and [13] have shown the deleted estimator (3.12) to outperform the basic held-out estimator (3.11) on large training corpora, with both methods being inferior to Good–Turing estimate of (3.9).

An alternative extension to cross-validation is *leaving-one-out* method [55] for which the training corpus of size N is split into a training part consisting of $(N-1)$ tokens and the held-out part consisting of only one token which is then used for a sort of simulated testing. This process is repeated N times so that all N tokens are used as the held-out part. The advantage of this training approach is that all N tokens are used both for the training and the held-out part, efficiently exploiting the whole training corpus. In particular, this method explores the effect of how the model changes if any particular token had not been observed.

Both deleted estimation and leaving-one-out approaches can be shown to lead to Good–Turing estimates [33] [58]. Derivation of a Good–Turing estimate using leaving-one-out method will be discussed in Sect.3.1.5. First, the unconstrained discounting model is presented.

3.1.4 Unconstrained discounting model

As it was shown in Sect.3.1.2, applying Turing estimate (3.9) is not a feasible solution because of the problem of high-frequency types. An alternative to original Good's smoothing solution is to use Good–Turing re-estimation only for frequencies $r < k$ for some k. Low frequency words are quite numerous, so substitution of the observed r for the expectations will be quite an accurate approximation, while the regular MLE estimates for high frequencies will be accurate without need for discounting.

An unconstrained model for discounting can also be constructed heuristically without need to resort to Good–Turing analysis. By letting k be the maximum count and λ_r the count dependent discounting factors for $r < k$, given the normalisation constraint the gained probability mass has to be redistributed over the unseen events. Thus following model for *joint* probability of events (h, w) is proposed in

[58]

$$P(h,w) = \begin{cases} \dfrac{N(h,w)}{N} & N(h,w) \geq k \\ \left[1 - \lambda_{N(h,w)}\right] \dfrac{N(h,w)}{N} & 0 < N(h,w) < k \\ \left(\displaystyle\sum_{h'w':0<N(h',w')<k} \left[\lambda_{N(h',w')} \frac{N(h',w')}{N} \right] \right) \dfrac{1}{n_0} & N(h,w) = 0 \ , \end{cases}$$

where n_0 is the estimated number of unseen events defined in (3.10). By examining the previous definition and somewhat simplifying the count notation following equations are obtained

$$\begin{aligned} \sum_{h,w} N(h,w) &= \sum_{r=0}^{k} r n_r = \sum_{r=1}^{k} r n_r = N \ , \\ \sum_{hw:0<N(h,w)<k} \lambda_{N(h,w)} \frac{N(h,w)}{N} &= \sum_{r=1}^{k-1} \lambda_r \frac{r n_r}{N} \ . \end{aligned} \tag{3.13}$$

Equations in (3.13) will be useful for deriving the closed form solution for optimal discounting factors λ_r in the next section.

3.1.5 Leaving-one-out estimate for joint probabilities

The N events are the *joint* events (h, w) obtained from the training corpus $\mathcal{T}$, where

$$\mathcal{T} : w_1, \ldots, w_i, \ldots, w_N \ ,$$

by isolating the word w_i and its history h_i in each of the N text positions. Consider the process of removing a certain observation $(h_i, w_i) = (h, w)$ from N observations and using it as a held-out part. Let the original count be $r = N(h, w)$. After removing one observation there are $(r - 1)$ observations of the same type left in $(N - 1)$ training observations. Therefore, for the held-out part of the data, count $(r - 1)$ and discounting factor λ_{r-1} must be used. From (3.13) it follows that there are $r n_r$ observations used as held-out data with parameter λ_{r-1}. Therefore by summing up over all the counts $r \in [1, k]$, the full log-likelihood function $\tilde{F}(\{\lambda_r\})$ of leaving-one-out method is defined as

$$\begin{aligned} \tilde{F}(\{\lambda_r\}) &= \sum_{h,w} N(h,w) \log P(h,w) \\ &= \sum_{hw:N(h,w)=1} 1 \cdot \log P(h,w) + \sum_{hw:1<N(h,w)<k} N(h,w) \log P(h,w) \\ &= \sum_{hw:N(h,w)=1} 1 \cdot \log \left[\frac{1}{n_0} \sum_{r=1}^{k-1} \lambda_r \frac{r n_r}{N-1} \right] + \\ &\quad \sum_{hw:1<N(h,w)<k} N(h,w) \log \left(\left[1 - \lambda_{N(h,w)-1}\right] \frac{N(h,w)-1}{N-1} \right) \ . \end{aligned}$$

The decomposition of log-likelihood function $\tilde{F}(\cdot)$ into two parts (one part for all events with $r = 1$ and another for all events with $1 < r < k$) is essential, since

after holding out one count, the form of probability function for all the original singleton events becomes the form defined for unseen events while the second part of decomposition reflects the probability for the rest of the counts which stay non-zero. Partial log-likelihood $F(\{\lambda_r\})$ which is used to find the optimal values of $\{\lambda_r\}$ is constructed by considering only the λ_r-dependent parts of $\tilde{F}(\cdot)$, *i.e.*

$$\begin{aligned}
F(\{\lambda_r\}) &= \sum_{hw:N(h,w)=1} \log\left[\sum_{r=1}^{k-1} \lambda_r r n_r\right] + \\
&\quad \sum_{hw:1<N(h,w)<k} N(h,w) \log\left[1-\lambda_{N(h,w)-1}\right] \\
&= \sum_{hw:N(h,w)=1} \log\left[\sum_{r=1}^{k-1} \lambda_r r n_r\right] + \\
&\quad \sum_{r=2}^{k} \left(\sum_{hw:N(h,w)=r} r \log\left[1-\lambda_{r-1}\right] \right) \\
&= n_1 \log\left[\sum_{r=1}^{k-1} \lambda_r r n_r\right] + \sum_{r=2}^{k} r n_r \log\left[1-\lambda_{r-1}\right] \\
&= n_1 \log\left[\sum_{r=1}^{k-1} \lambda_r r n_r\right] + \sum_{r=1}^{k-1} (r+1) n_{r+1} \log\left[1-\lambda_r\right] \quad , \qquad (3.14)
\end{aligned}$$

where the definitions from (3.13) have been used. Taking partial derivatives with respect to each λ_r where $r \in [1, k-1]$ in (3.14) and equating them to zero, following system of $(k-1)$ equations for $(k-1)$ unknown parameters is obtained

$$\frac{\partial F}{\partial \lambda_r} = n_1 \frac{r n_r}{\sum_{s=1}^{k-1} \lambda_s s n_s} - \frac{(r+1) n_{r+1}}{1-\lambda_r} = 0 \ . \qquad (3.15)$$

By exploiting the fact that the sum in (3.15) does not depend on r index of λ_r to be found, following closed-form solution is obtained

$$\begin{aligned}
\lambda_r &= 1 - \left[\frac{\sum_{s=1}^{k-1} s n_s}{\sum_{s=1}^{k} s n_s}\right] \frac{(r+1) n_{r+1}}{r n_r} \\
&= 1 - \left[1 - \frac{k n_k}{N}\right] \frac{(r+1) n_{r+1}}{r n_r} \ . \qquad (3.16)
\end{aligned}$$

Therefore, by plugging in λ_r from (3.16) for the events (h, w) with $r \neq k$ into the unconstrained discounting model, following probability estimates are obtained

$$r \neq k : P_{\mathrm{LOO}}(N(h,w) = r) = \left[1 - \frac{k n_k}{N}\right] \frac{(r+1) n_{r+1}}{N n_r} \ . \qquad (3.17)$$

The probability mass of unseen events therefore is

$$\sum_{hw:N(h,w)=0} P_{\mathrm{LOO}}(h,w) = \left[1 - \frac{k n_k}{N}\right] \frac{n_1}{N}$$

and the total probability mass of events that were seen in training $(r > 0)$ is given by

$$\sum_{hw:N(h,w)=r} P_{\mathrm{LOO}}(h,w) = \left[1 - \frac{k n_k}{N}\right] \frac{(r+1) n_{r+1}}{N} \ .$$

Given the leaving-one-out probability estimate in (3.17), ignoring the probability mass of counts $r = k$ and assuming

$$\frac{kn_k}{N} \ll 1$$

we obtain Good–Turing estimate

$$P_{\mathrm{GT}}(h, w) \approx \frac{1}{N} \frac{(r+1)n_{r+1}}{n_r} \ ,$$

from which the following Good–Turing discounting factor is obtained

$$\lambda_r \approx 1 - \frac{(r+1)n_{r+1}}{rn_r}$$

and the Good–Turing estimate of the probability mass of the unseen events becomes

$$\sum_{hw:N(h,w)=0} P_{\mathrm{GT}}(h, w) \approx \frac{n_1}{N} \ , \tag{3.18}$$

which is a very useful equation for checking the coverage of the vocabulary.

3.2 Combination of Estimators

All of the techniques described so far make use of raw count r of an n-gram as a base for prediction. These methods assign the same probability for all n-grams that never appeared or appeared only rarely, which is not desirable. In theory we would like to estimate different probabilities for n-grams with the same frequency in order to account better for occurrences of different word patterns in natural language. For example, in such cases, one might hope to produce better estimates by looking at the frequency of $(n-1)$-grams found in n-gram. This will supply additional information which then can be used to refine estimates. One of the initial developments in this area is described in [17] where the estimates for the unseen bigrams are shown to be refined in terms of probabilities of unigrams that compose them using the bins describing disjoint groups, with each bin treated as a separate distribution and Good–Turing estimation is performed on each, giving corrected counts which are normalised to yield probabilities.

In this section the problem of combining probability estimates from different models is presented and several popular solutions are described.

3.2.1 Katz's backing-off

Katz smoothing [37] extends the intuitions of Good–Turing estimate by adding combination of different models which are consulted in order depending on their specificity. The most detailed model that is able to provide sufficiently reliable information about the current context is used and models are defined recursively in terms of lower order models.

Using Katz's unconstrained model, the count-dependent discounting factors λ_r are applied only for $r \leq k$, where k is some integer constant (Katz suggests $k = 5$). To satisfy the normalisation constraint, the gained probability mass has to be computed separately for each history and then distributed over the unseen events using a more general (lower-order) distribution $Q(\cdot)$ in the process of *backing-off*.

A generalised history $\widehat{h}$ for the n-gram (h, w) is defined as $(n-1)$-gram $(\widehat{h}, w)$ (for example, a bigram history of a trigram $(h, w) = (w_1, w_2, w_3)$ would have a unigram $(\widehat{h}, w) = (w_2, w_3)$ as generalised history). Therefore a more general distribution $Q(w|\widehat{h})$ is conditioned on generalised history $\widehat{h}$.

In Katz's model the estimate for conditional probability is defined as

$$P(w|h) = \begin{cases} \dfrac{N(h,w)}{N(h)} \\ \qquad N(h,w) > k \\ \left[1 - \lambda_{N(h,w)}\right] \dfrac{N(h,w)}{N(h)} \\ \qquad 1 \leq N(h,w) \leq k \\ \left(\displaystyle\sum_{w':0<N(h,w')\leq k} \left[\lambda_{N(h,w')} \frac{N(h,w')}{N(h)} \right] \right) \dfrac{Q(w|\widehat{h})}{\displaystyle\sum_{w':N(h,w')=0} Q(w'|\widehat{h})} \\ \qquad N(h,w) = 0 \ , \end{cases}$$

which is subject to usual normalisation constraint $\sum_w P(w|h) = 1$. The large counts $N(h, w) > k$ are taken to be reliable and are not discounted.

Using the following count-counts

$$\begin{aligned} n_r(h) &\triangleq \sum_{w:N(h,w)=r} 1 \\ n_r &\triangleq \sum_h n_r(h) \end{aligned}$$

for which

$$\sum_{w:1\leq N(h,w)\leq k} \lambda_{N(h,w)} \frac{N(h,w)}{N(h)} = \sum_{r=1}^{k} \lambda_r \frac{r n_r(h)}{N(h)} \ ,$$

leaving-one-out method [58] constructs the log-likelihood function in a way similar to section 3.1.5 in order to obtain optimal values for discounting coefficients λ_r arriving at the following closed-form solution

$$\lambda_r = 1 - \left[\frac{\sum_{s=1}^{k} s n_s}{\sum_{s=1}^{k+1} s n_s} \right] \frac{(r+1) n_{r+1}}{r n_r} \ , \tag{3.19}$$

which is virtually identical to leaving-one-out discounting in the case of joint probabilities from (3.16).

The original solution for λ_r [37] requires the total probability mass of unseen events to equal the Good–Turing estimate for unseen events from (3.18), *i.e.*

$$\sum_{r=1}^{k} \lambda_r \frac{r n_r}{N} = \frac{n_1}{N}$$

and λ_r are defined by re-normalising the corresponding Good–Turing factors with a factor μ

$$\lambda_r = \mu \left[1 - \frac{(r+1) n_{r+1}}{r n_r} \right] \approx \mu \left[1 - \frac{r^*}{r} \right] \ ,$$

obtaining the final estimates

$$\lambda_r = \frac{1 - \dfrac{(r+1)n_{r+1}}{rn_r}}{1 - \dfrac{(k+1)n_{k+1}}{n_1}} \approx \frac{1 - \dfrac{r^*}{r}}{1 - \dfrac{(k+1)n_{k+1}}{n_1}} \quad . \tag{3.20}$$

Note that the developments leading to derivations of (3.19) and (3.20) given the Katz's model so far have assumed that generalised distribution $Q(w|\widehat{h})$ is known. It is obvious, however, that there is needed some way of estimating the generalised distribution itself. For the standard model [37], the relative frequency estimates of lower-order events are advocated, *i.e.*

$$Q(w|\widehat{h}) = \frac{N(\widehat{h}, w)}{\sum_{w'} N(\widehat{h}, w')} \quad . \tag{3.21}$$

This move is justified since the estimate in (3.21) is a conventional maximum likelihood estimate.

3.2.2 Linear discounting

In linear discounting [55] [57] the non-zero maximum likelihood estimates are scaled by a constant slightly less than one and the remaining probability mass is redistributed across novel events

$$P(w|h) = \begin{cases} (1-\alpha)\dfrac{N(h,w)}{N(h)} & N(h,w) > 0 \\ \alpha \dfrac{Q(w|\widehat{h})}{\sum_{w':N(h,w')=0} Q(w'|\widehat{h})} & N(h,w) = 0 \quad . \end{cases}$$

In general, leaving-one-out method is used in [57] to obtain a following closed-form solution for α

$$\alpha = \frac{n_1}{N} \quad ,$$

which is identical to Good–Turing probability estimate for the unseen events.

This model has been shown to perform rather poorly [56] [50] since both high and low frequencies are discounted by the same constant factor but it is well known that the higher the frequency, the more accurate a raw maximum likelihood estimate is, property which is not reflected by linear discounting.

3.2.3 Absolute discounting

Absolute discounting method [57] attempts to leave the non-zero counts virtually unchanged. The heuristic justification for this may be found in [58] where it is argued that the count r for a certain event (h, w) does not change significantly with the replacement of the training corpus of size N with another corpus of the same size and the variation of r can be expected to be in range $[r-1, r+1]$. This leads to introduction of average (non-integer) count offset b independent of the count r itself. Subject to normalisation constraint, the model thus takes the following form

$$P(w|h) = \begin{cases} \dfrac{N(h,w) - b}{N(h)} & N(h,w) > 0 \\ b\dfrac{S - n_0(h)}{N(h)} \dfrac{Q(w|\widehat{h})}{\sum_{w':N(h,w')=0} Q(w'|\widehat{h})} & N(h,w) = 0 \quad , \end{cases}$$

where the constraint on b is $0 < b < 1$ and S is the size of the universe of the events.

Using leaving-one-out estimation no closed-form solution can be obtained and the approximation of an upper bound on b is found to be

$$b_{\mathrm{LOO}} = \frac{n_1}{n_1 + 2n_2} \ .$$

Using the approach taken in [58], Good–Turing probability mass for unseen events leads to another possible estimate of b

$$b_{\mathrm{GT}} = \frac{n_1}{\sum_{r \geq 1} n_r} \ .$$

Several enhancements to absolute discounting are possible. In [58] usage of two discounting parameters (one for singleton events with $r = 1$ and another for $r > 1$) is advocated which leads to improved performance. Additional refinement suggested is to apply an absolute discounting only to low counts, similar to Katz model, and use MLE estimates for high frequencies.

3.2.4 Kneser-Ney smoothing

In the approaches presented so far, the lower-order distribution is taken to be a smoothed version of the lower-order maximum likelihood distribution. However, the lower-order distribution $Q(w|\widehat{h})$ becomes an important factor only when there are few or no counts present in the higher-order distribution. Method presented in this section tries to simulate this condition.

Kneser and Ney [40] note that for word bigrams such as *bona fide* or *Sri Lanka* (and many other collocations [18] and proper names [52]) the second word is strongly coupled with the first one and thus its unigram count will be high if its predecessor word occurs often in the corpus. But in backing-off case we know exactly that a predecessor word has not occurred. As a result, the relative frequencies of word unigrams over-estimate the true probabilities. To solve this problem, authors propose to use the generalised singleton distribution, *i.e.* computing lower-order distribution only from those word bigrams that were seen only once

$$Q(w|\widehat{h}) \approx \frac{N_1(\widehat{h}, w)}{\sum_{w'} N_1(\widehat{h}, w')} \ ,$$

where

$$N_1(\widehat{h}, w) = \sum_{h \in \widehat{h}: N(h,w)=1} 1 \ .$$

Such smoothing can be applied to any of the backing-off methods discussed above and its derivation is described in detail in [40] and [58]. Several experiments with language models improved by incorporating a singleton backing-off distribution are described in [25].

In recent experiments [14] [15] Kneser-Ney smoothing and its variants (some of them based on interpolation) were found to consistently outperform all other approaches.

3.2.5 Linear interpolation

An important alternative to back-off models described so far is linear interpolation technique, in which higher-order models are *mixed* with lower-order models that

suffer less from data sparseness. The basic form of linear interpolation can be obtained from *floor method* [55] related to Lidstone's law (3.5), for which instead of adding a constant floor value, some value proportional to a less specific distribution $Q(\cdot)$ is added, *i.e.*

$$P_{\text{Floor}}(h,w) = \frac{N(h,w) + \delta_h Q(\widehat{h},w)}{N(h) + \delta_h} \ , \tag{3.22}$$

where δ_h is dependent on history. By introducing a new constant λ'_h for which

$$\lambda'_h = \frac{\delta_h}{N(h) + \delta_h}, \quad 0 < \lambda'_h \leq 1$$

equation (3.22) becomes the linear interpolation formula proposed by Jelinek [31] [32]

$$P_{\text{LI}}(h,w) = (1 - \lambda'_h)\frac{N(h,w)}{N(h)} + \lambda'_h Q(\widehat{h},w) \ ,$$

which for conditional probabilities may be rewritten as

$$P_{\text{LI}}(w|h) = (1 - \lambda_h)\frac{N(h,w)}{N(h)} + \lambda_h Q(w|\widehat{h}) \ . \tag{3.23}$$

An alternative way to arrive to the same result (3.23) is to replace concept of backed-off linear discounting described in Sect.3.2.2 by interpolation [57].

The reason for making the interpolation parameters λ_h history-dependent becomes apparent upon considering a fact that for higher counts the higher distribution is more reliable, therefore smaller value of λ in (3.23) will be appropriate, whereas for low counts setting a larger λ is desirable.

The smoothing parameters λ_h are chosen to maximise the probability estimate $P_{\text{LI}}(\cdot)$ in (3.23). The solution advocated in [31] is to use Baum-Welsh (EM) algorithm which is guaranteed to converge to a local optimum. This method approaches a language model smoothing problem from a rigorous Hidden Markov Model (HMM) point of view [3] [31] [11] and exploits the training corpus in a process of *deleted interpolation*, which is similar to deleted estimation method of Sect.3.1.3.

An alternative derivation of estimation equations for linear interpolation is given in [57], where authors use the leaving-one-out formalism to arrive at formulae which happen to produce the correct iteration equations, but without the convergence guarantee of Baum-Welsh algorithm. In this case the re-estimation equation for interpolation parameter is given by

$$\widehat{\lambda}_h = \frac{1}{N(h)}\left[n_1(h) + \sum_{w:N(h,w)>1} N(h,w)\frac{\lambda_h Q(w|\widehat{h})}{(1-\lambda_h)\dfrac{N(h,w)-1}{N(h)-1} + \lambda_h Q(w|\widehat{h})} \right] \ ,$$

where the influence of singleton distribution $n_1(h)$ is evident.

3.2.6 Unified view on backing-off and linear interpolation

As noted in [40], most existing smoothing methods can be described by the following back-off equation

$$P_{\text{BO}}(w|h) = \begin{cases} \alpha(w|h) & N(h,w) > 0 \\ \gamma(h) Q(w|\widehat{h}) & N(h,w) = 0 \ , \end{cases}$$

where $\alpha(w|h)$ is some reliable estimate of probability (*e.g.* ML), $Q(w|\widehat{h})$ is a less specific distribution and $\gamma(h)$ is the scaling factor determined completely by $\alpha(\cdot)$ and $Q(\cdot)$.

Another group of smoothing methods is expressed as linear interpolation of higher and lower-order models and is given by (3.23), which can be rewritten as

$$P_{\mathrm{LI}}(w|h) = \alpha'(w|h) + \gamma(h)Q(w|\widehat{h}) \ ,$$

where

$$\alpha'(w|h) = (1-\lambda_h)\frac{N(h,w)}{N(h)}$$

and $\gamma(h) = \lambda_h$. By taking

$$\alpha(w|h) = \alpha'(w|h) + \gamma(h)Q(w|\widehat{h}) \tag{3.24}$$

equation (3.24) can be placed in the above back-off form.

A common approach to place the interpolated model into the back-off framework is to take

$$\alpha(w|h) = \alpha'(w|h)$$

and adjust the normalisation factor $\gamma(h)$ so that probabilities sum to one.

3.2.7 Parameter tying

In order to reduce the number of free history-dependent parameters λ_h in the context of linear interpolation they need to be pooled across different histories. Placing the interpolation model in back-off context, as described in previous section, we obtain

$$P(w|h) = \begin{cases} (1-\lambda_h)\dfrac{N(h,w)}{N(h)} & N(h,w) > 0 \\ \lambda_h \left[\dfrac{Q(w|\widehat{h})}{\sum_{w':N(h,w)=0} Q(w'|\widehat{h})}\right] & N(h,w) = 0 \ . \end{cases}$$

Using leaving-one-out formalism following possible estimates are obtained by Ney [58] for different types of tying:

No tying: For each history h,

$$\lambda_h = \frac{n_1(h)}{N(h)} \ .$$

History count tying: The assumption is that the parameters λ_h depend on history h only via the history count $N(h)$, *i.e.* for $r \in \{N(h) : h\}$

$$\lambda_r = \frac{\sum_{h:N(h)=r} n_1(h)}{\sum_{h:N(h)=r} N(h)} \ .$$

Often the tying process is carried further by dividing the history counts into moderate number of partitions or *bins* and using separate parameter λ for each bin [34], approach taken in the experiments described below. Extension to this approach, namely the *average-count* method proposed by Chen [13], for which parameters are tied according to the average number of counts per non-zero element in the history, was reported to yield an even better performance.

No history dependence: Results in linear discounting model of Sect.3.2.2 and

$$\lambda = \frac{\sum_h n_1(h)}{\sum_h N(h)} = \frac{n_1}{N} \ .$$

Full (h, w)-tuple dependence: Results in Katz's model of Sect.3.2.1.

As it can be seen, different types of tying result in different smoothing models. In general, having too many interpolation paramaters is not desirable, since this defeats purpose of cross-validation which in this case does not differ from conventional learning [58]. It is therefore recommended to use reasonable amount of tying to improve the robustness of the model with respect to the new data.

CHAPTER 4

Interpolation of Language Models

This chapter presents the theoretical framework for linear and log-linear interpolation, which are the two different ways of combining any n-gram probability estimates. In particular, in the scope of this thesis, we are interested in developing the frameworks for linear and log-linear interpolation of the n-gram language models of the same nature (*i.e.* built using the same concept and text corpus) but with varying degree of specificity. Efficient ways of probability estimation, parameter optimisation and tying are presented, along with theoretical comparative analysis of both techniques. Section 4.1 presents the linear interpolation while section 4.2 presents the log-linear interpolation.

4.1 Linear Smoothing

Linear smoothing of probability estimates, known in statistical NLP as *linear interpolation* and as *mixture models* elsewhere, has been probably the most widely used technique for combining the language models. Linear interpolation has been briefly introduced in the previous chapter in the general framework of language modelling smoothing. In this section it is discussed in more detail.

The most basic way to linearly combine the m probability estimates is to take

$$P_{\mathrm{LI}}(w|h) = \sum_{k=1}^{m} \lambda_k P_k(w|h) \ , \tag{4.1}$$

where

$$\forall k \in [1,m]: \quad 0 \leq \lambda_k \leq 1, \quad \sum_k \lambda_k = 1 \ .$$

Linear interpolation cannot hurt. The optimally interpolated model given by (4.1) is guaranteed to be no worse than any of its components. This is because each of the components can be viewed as a special case of the interpolation, with a weight of 1 for that component and 0 for the others. This is only guaranteed for a held-out data, not for a new data. But if the held-out data is large enough and representative, the result will carry over to the test data as well.

Since this is a general technique, it has been frequently applied to combining the stochastic models of different nature. Some of its recent applications include combination of parser models [12], topic-dependent models [71], back-off and maximum entropy models [51] and interpolation of cache, Kneser-Ney smoothed, high-order n-gram, skipping and sentence-based models [28].

Within the context of this paper, we are concerned with linearly combining probability estimates $P(\cdot)$ of the n-gram models obtained from the same corpus. Expressed in terms of the n-grams, the estimate in (4.1) can be expressed as

$$P_{\text{LI}}(w_i|w_{i-n+1}^{i-1}) = \lambda_1 P(w_i|w_{i-n+1}^{i-1}) + \ldots + \lambda_n P(w_i) \tag{4.2}$$

with the same constraints imposed for the interpolation weights λ_k. By assuming the history-dependence of the parameters, introduced in Sect.3.2.5, an alternative functional form to that of (4.2) is to define the linear interpolation recursively [13] in the following way

$$\begin{aligned} P_{\text{LI}}(w_i|w_{i-n+1}^{i-1}) &= (1-\lambda_{w_{i-n+1}^{i-1}})P(w_i|w_{i-n+1}^{i-1}) + \\ &\quad \lambda_{w_{i-n+1}^{i-1}} P_{\text{LI}}(w_i|w_{i-n+2}^{i-1}) \ . \end{aligned} \tag{4.3}$$

This allows us to define a smoothed n-gram model recursively as linear interpolation between the nth-order actual probability estimate and $(n-1)$th-order smoothed model. Recursion is terminated by either taking the 1st-order smoothed model to be a unigram probability estimate as it is done in this work,

$$P_{\text{LI}}(w_i|w_{i-1}) = (1-\lambda_{w_{i-1}})P(w_i|w_{i-1}) + \lambda_{w_{i-1}} P(w_i)$$

or by taking the 0th-order smoothed model to be a uniform distribution (zerogram) as follows

$$P_{\text{LI}}(w_i) = (1-\lambda_0)P(w_i) + \lambda_0 \frac{1}{|\mathcal{V}|} \ ,$$

where $|\mathcal{V}|$ is the vocabulary size. This approach was taken by Brown [10] and Chen [13]. It remains to be shown which one of the two approaches is more advantageous.

4.1.1 Parameter tying

As it was briefly mentioned before, from an intuitive point of view, parameters λ_h should be different for different histories h. If a context h was seen sufficient number of times to reliably estimate the parameter this should be reflected by an appropriately low value of λ_h making the contribution of a particular probability distribution $P(\cdot)$ in (4.3) more significant and, analogously, if there was not enough times context h had appeared during the training, λ_h should be higher, giving more probability mass to the lower-order smoothed model.

It is not generally feasible to accurately train each interpolation parameter λ_h independently since an enormous amount of data would be needed for such an estimation. Jelinek [31] suggested to divide the parameters λ_h into the moderate number of bins, constraining all the parameters λ_h in the same bin to have the same value. The mapping between the history $h = w_{i-n+1}^{i-1}$ and a bin $\mathcal{B}_k$ is via the number of times this history has appeared in the training corpus, *i.e.*

$$\forall h \in \mathcal{T} : \ \text{if} \ \ N(h) \in \mathcal{B}_k \ \ \Rightarrow \ \ \lambda_h = \lambda(\mathcal{B}_k), \ \ \mathcal{B}_k \in \mathfrak{B} \ ,$$

where $\mathfrak{B}$ is the collection of all possible bins. Therefore the number of interpolation parameters is equal to the total number of bins. Intuitively, each bin $\mathcal{B}_k$ should be made as small as possible to only cluster together the most similar n-grams while remaining large enough to accurately estimate the corresponding parameter $\lambda(\mathcal{B}_k)$.

Bins are built using the clustering method often referred to as *wall of bricks* [47] in which bins are created so that each bin contains at least $N_{\min}$ n-gram history

counts. By starting from the minimal possible value of $N(h)$ and assigning the increasing values of $N(h)$ to this bin (updating the bin boundaries appropriately) the process continues until the minimum count $N_{\min}$ is reached, at which point process is repeated for the remaining history counts until all the possible values of $N(h)$ are clustered (*i.e.* assigned to corresponding bins). The clustering is done separately for all the n-gram models, yielding n different bin spaces $\mathfrak{B}$.

Typically, for histories with low counts, the bins will contain a large number of histories, whereas for higher history count values each bin will contain a small number of histories.

4.1.2 Parameter optimisation

The method adopted in this work for the purpose of describing the training and optimisation framework for linear interpolation is the *held-out estimation*, briefly described in Sect.3.1.3, in which one reserves a section of a training data, called the *kept* or *retained* data which is used for obtaining the n-gram probability estimates and determining the clustering parameters, while the remaining (usually much smaller) part of the training data, the held-out data, is used for optimising the interpolation weights and, in general, simulate the unseen test corpora. In alternative method, called *deleted interpolation* or *deleted estimation*, different parts of training data rotate simulating either the retained or held-out part and the results are then combined together [1].

Let $\mathcal{K}_{\mathcal{T}}$ denote the kept, and $\mathcal{H}_{\mathcal{T}}$ the held-out parts of the training corpus $\mathcal{T}$ respectively, with the following being true for the sizes of the corpora

$$|\mathcal{K}_{\mathcal{T}}| + |\mathcal{H}_{\mathcal{T}}| = |\mathcal{T}|$$

and

$$\delta \equiv \frac{|\mathcal{H}_{\mathcal{T}}|}{|\mathcal{K}_{\mathcal{T}}|} < 1 \ ,$$

where δ may be defined either intuitively, by corresponding to one's notion of what a kept and a held-out corpora size should be (the kept part should be big enough to reliably estimate the probabilities and the clustering parameters while the held-out part should be reasonably representative of the possible unseen test corpora and have a size which is sufficient for accurate parameter optimisation) or experimentally, by dividing the total training data $\mathcal{T}$ into the kept and held-out sets in such a way as to achieve the smallest perplexity of the model with respect to the second held-out set which is disjoint from the overall training data[2]. For instance, Chen and Goodman [14] [15] divide the overall training data into one kept set and two held-out sets, with the first held-out set being used for parameter optimisation while the second held-out set is solely used for cross-entropy calculation.

Using the recursive formalism defined in (4.3) it can be noted that it is sufficient to estimate the weight λ_h corresponding to history h independently for each bin $\mathcal{B}_k$ into which the history count $N(h)$ falls. Given the bins $\mathcal{B}_k \in \mathfrak{B}$ estimated using the kept part of the training corpus $\mathcal{K}_{\mathcal{T}}$, the interpolation parameter optimisation is carried out on the held-out corpus $\mathcal{H}_{\mathcal{T}}$ as shown in Alg.4.1. The per-bin log-likelihood function to be optimised is defined as

$$\mathcal{L}(\mathcal{B}_{n,k}) = \sum_{h:C(h)\in\mathcal{B}_{n,k}} \sum_{w} C(h,w) \log \left[(1-\lambda) P(w|h) + \lambda P_{\mathrm{LI}}(w|\widehat{h}) \right] \ , \qquad (4.4)$$

[1] Leaving-out-one can be seen as a special case of deleted estimation

[2] The choice of $\delta = \frac{6}{7}$ proved to be reasonable.

for all n-gram history orders n starting with bigrams ($n = 1$) **do**
  for all bins $\mathcal{B}_{n,k}$ in a collection $\mathfrak{B}_n$ **do**
    Find $\lambda(\mathcal{B}_{n,k})$ maximising the log-likelihood $\mathcal{L}(\mathcal{B}_{n,k})$ defined by (4.4).
  end for
end for

Algorithm 4.1: Log-likelihood optimisation for all the clusters (bins).

where $C(\cdot)$ denotes the counts obtained from the held-out set[3] and $\widehat{h}$ is a generalised ($(n-1)$th-order) history. Notation was simplified slightly by denoting the bin-dependent weight $\lambda(\mathcal{B}_{n,k})$ by λ. Summation in (4.4) is carried for all the events w in all the contexts h in held-out set $\mathcal{H}_\mathcal{T}$ such that their history counts fall into the bin $\mathcal{B}_{n,k}$, which is defined by the kept set $\mathcal{K}_\mathcal{T}$.

The log-likelihood function $\mathcal{L}(\mathcal{B}_{n,k})$ is optimised in a bottom-up fashion in terms of n-gram history orders. For the initial (bigram) case $P_{\mathrm{LI}}(w|\widehat{h})$ is equivalent to a unigram probability estimate $P(w)$, whereas for the higher-order histories $n > 1$ it is equal to the lower-order linearly smoothed estimate.

It can be shown for the special case when the probability distributions being smoothed are maximum likelihood estimates, that optimising the log-likelihood function in (4.4) with respect to the kept set is, in fact, equivalent to assigning maximum weight, *i.e.* $\lambda(\mathcal{B}_{n,k}) = 0$, to the higher-order maximum likelihood distribution [33], since in this case the procedure is essentially similar to the one leading to the maximum likelihood estimation, as shown in Sect.2.2.1, maximising the log-likelihood of a model with respect to itself.

Log-likelihood function $\mathcal{L}(\mathcal{B}_{n,k})$ can be proven to represent a convex function of λ in the range $0 \leq \lambda \leq 1$ and from this property it immediately follows that it has a unique local maximum in this range. Consequently, since log-likelihood is a monotonic function, the likelihood function also has a unique local maximum in this range. Taking the derivative of (4.4) with respect to λ, following expression is obtained

$$\frac{d}{d\lambda}\mathcal{L}(\mathcal{B}_{n,k}) = \sum_{h:C(h)\in\mathcal{B}_{n,k}} \sum_{w} C(h,w) \left[\lambda + \frac{P_{\mathrm{LI}}(w|\widehat{h})}{P_{\mathrm{ML}}(w|h) - P_{\mathrm{LI}}(w|\widehat{h})}\right]^{-1} . \qquad (4.5)$$

Following from the property of convex functions, the log-likelihood function must attain its maximum value at either one of the boundary points $\lambda = 0$ (in which case this is a monotonically decreasing function in this range) or $\lambda = 1$ (in which case the function monotonically increases in the range), or at any interior point $0 < \lambda < 1$ (in which case the function monotonically increases from 0 to λ_{opt} and then monotonically decreases to 1), Bahl *et al.* [2] present a fast algorithm for finding the optimal value λ_{opt} of λ by searching for the root of the derivative of log-likelihood function given by (4.5). Since second derivative given by

$$\frac{d^2}{d\lambda^2}\mathcal{L}(\mathcal{B}_{n,k}) = \sum_{h:C(h)\in\mathcal{B}_{n,k}} \sum_{w} C(h,w) \left[\lambda + \frac{P_{\mathrm{LI}}(w|\widehat{h})}{P_{\mathrm{ML}}(w|h) - P_{\mathrm{LI}}(w|\widehat{h})}\right]^{-2}$$

is always non-positive, this solution is guaranteed to be local maximum. The algorithm, which employs technique for dividing the search interval the authors call

[3] Recall that $N(\cdot)$ refer to counts from the kept set.

Require: $n_{\max} = -\log_2 \epsilon - 1$
where $\epsilon = 2^{-(n_{\max}+1)}$ is the required tolerance.
if $D(0) \leq 0$ **then**
$\lambda_{\text{opt}} = 0$. Stop.
else if $D(1) \geq 0$ **then**
$\lambda_{\text{opt}} = 1$. Stop.
end if
$l = 0$, $r = 1$
for $n = 0$ to $n_{\max}$ **do**
$m = (l + r)/2$
if $n = n_{\max}$ or $D(m) = 0$ **then**
$\lambda_{\text{opt}} = m$. Stop.
else if $D(m) > 0$ **then**
$l = m$
else if $D(m) > 0$ **then**
$r = m$
end if
end for

Algorithm 4.2: Interval search for an optimal λ.

binary chopping search, shown[4] in Alg.4.2, was reported to be one or two orders of magnitude faster than the standard forward-backward (EM) algorithm (employed, for instance, by Peto [61] and Rosenfeld [64]) and can be thought of as a faster performing variant of bisection method for root finding, with the only difference being that no initial bracketing constraints on the root are imposed, since the behaviour of the function is initially known. Additional improvements in performance which can be obtained by using the Newton-Raphson method with second derivatives [62], were not attempted since performance of an algorithm seemed to be satisfactory enough.

4.2 Log-Linear Smoothing

An alternative way of combining the language models may be derived using the following framework which exploits the constrained conditional relative entropy approach. Given the n probability distributions $P_i(w|h)$ to be combined, $i \in [1, n]$, the conditional relative entropy of the unknown target model $P(w|h)$ with respect to each of the given models is defined by the following non-symmetric Kullback-Leibler distance measure

$$D(P(w|h)\|P_i(w|h)) = \sum_h P(h) \sum_w P(w|h) \log \frac{P(w|h)}{P_i(w|h)} = d_i \ ,$$

where $D(\cdot)$ is a relative entropy between conditional probability distributions $P(w|h)$ and $P_i(w_h)$ and d_i are the constraints on the system. The target probability distribution should be minimised in terms of its conditional relative entropy with respect to some additional model $P_0(w|h)$. Using Lagrangian multipliers γ_i the constrained

[4]The notation is simplified by denoting $\frac{d}{d\lambda}\mathcal{L}(\mathcal{B}_{n,k})$ by $D(\lambda)$.

system function $\mathcal{D}_\Gamma(P(w|h))$ to be minimised can be expressed as

$$\mathcal{D}_\Gamma(P(w|h)) = D(P(w|h)\|P_0(w|h)) + \sum_i \gamma_i \left(D(P(w|h)\|P_i(w|h)) - d_i\right) \ ,$$

where $\Gamma = \{\gamma_1 \dots \gamma_n\}$. Taking partial derivatives with respect to γ_i and equating them to zero following equation is obtained

$$\frac{\partial \mathcal{D}_\Gamma}{\partial \gamma_i} = D(P(w|h)\|P_i(w|h)) - d_i = 0 \ .$$

Similarly, with respect to the target model,

$$\begin{aligned}\frac{\partial \mathcal{D}_\Gamma}{\partial P(w|h)} &= \sum_h P(h) \sum_w \left(1 + \log \frac{P(w|h)}{P_0(w|h)}\right) + \\ &\quad \sum_i \gamma_i \sum_h P(h) \sum_w \left(1 + \log \frac{P(w|h)}{P_i(w|h)}\right) = 0 \ .\end{aligned}$$

Rearranging the terms yields

$$\sum_h P(h) \sum_w \left[1 + \log \frac{P(w|h)}{P_0(w|h)} + \sum_i \gamma_i \left(1 + \log \frac{P(w|h)}{P_i(w|h)}\right)\right] = 0 \ . \tag{4.6}$$

Assuming that

$$\forall h \in \mathcal{V}: \quad P(h) \neq 0$$

and equating the inner sum of (4.6) to zero yields

$$(1 + \sum_i \gamma_i) \log P(w|h) + 1 - \log P_0(w|h) + \sum_i \gamma_i (1 - \log P_i(w|h)) = 0 \ . \tag{4.7}$$

Assuming P_0 to be a uniform distribution[5], taking the exponential and making some trivial rearrangements yields

$$P(w|h) = \exp\left(-1 - \frac{\log |\mathcal{V}|}{1 + \sum_i \gamma_i}\right) \prod_i P_i(w|h)^{\frac{\gamma_i}{1 + \sum_i \gamma_i}} \ . \tag{4.8}$$

Let

$$\lambda_i \triangleq \frac{\gamma_i}{1 + \sum_i \gamma_i}$$

denote the *interpolation weights*. The first term in the above product (4.8) is only dependent on the weights $\{\lambda_i\}$ and is denoted as $Z_0(\Lambda)$, which allows to express the target model as

$$P(w|h) = Z_0(\Lambda) \prod_i P_i(w|h)^{\lambda_i}, \quad \Lambda = \{\lambda_1 \dots \lambda_n\} \ . \tag{4.9}$$

In order for (4.9) to define a proper probability distribution it should be properly normalised. By introducing the history-dependent normalisation factor $Z_h(\Lambda)$ following equation is obtained

$$P_{\text{LLI}}(w|h) = \frac{1}{Z_h(\Lambda)} \prod_i P_i(w|h)^{\lambda_i} \ , \tag{4.10}$$

[5] *i.e.*, $P_0(w|h) = \frac{1}{|\mathcal{V}|}$, where $|\mathcal{V}|$ is the size of the vocabulary.

where $Z_0(\cdot)$ has been absorbed into the normalisation factor $Z_h(\Lambda)$ for which following is true

$$Z_h(\Lambda) = \sum_w \prod_i P_i(w|h)^{\lambda_i} \ ,$$

therefore obtaining

$$P_{\text{LLI}}(w|h) = \frac{\prod_i P_i(w|h)^{\lambda_i}}{\sum_w \prod_i P_i(w|h)^{\lambda_i}} \ .$$

Equation (4.10) has been introduced by Klakow [39] and can be viewed as linear interpolation in log domain, where in contrast with regular linear interpolation described in the preceding section, no explicit constraints appear on the log-linear interpolation weights.

As can be seen, the log-linear interpolation estimate of (4.10) is not reliable if the probability estimates to be smoothed are maximum likelihoods. In case where at least one of the events w was not observed in a certain context h, the corresponding maximum likelihood estimate becomes zero and the total log-linear estimate is assigned a zero value, which is undesirable since this constitutes a badly smoothed estimate. Additional problem, which may arise during the optimisation of the interpolation parameters, is when at least one of the interpolation parameters and its corresponding estimate are equal to zero, in which case the quantity in equation (4.10) becomes undefined. It is therefore desirable to use nonzero probability estimates for smoothing (*e.g.*, back-off estimates).

4.2.1 Formal framework

Following factors need to be considered in order to define an efficient framework for log-linear interpolation:

- The interpolation parameters Λ need to be made history dependent in order to better account for the data sparsity, the less sparse the back-off estimates for the history h, the bigger the corresponding parameter λ_h.
- Since interpolation parameters are history dependent, the normalisation factors are also history dependent, and, in addition, depend on the choice of the interpolation weights. However, because of the definition of the normalisation factor as being the summation of probability factors over all the events in the vocabulary given the specific history, no immediate way of clustering the normalisation factors is available, *i.e.* normalisation factors are strictly history-dependent, unlike the interpolation parameters which may be made cluster-dependent.
- Special handling is needed for the cases where the requested history is not found in the training set, in which case no normalisation factor nor one or more interpolation weights are available, in which case lower order linear interpolation model should be used.

The aforementioned considerations result in proposing the following framework for obtaining the reliable n-gram log-linear probability estimates,

$$P_{\text{LLI}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \left[Z_{w_{i-n+1}^{i-1}}(\Lambda)\right]^{-1} \prod_{j=1}^{n} P(w_i|w_{i-n+j}^{i-1})^{\lambda_j(w_{i-n+1}^{i-1})} & N(w_{i-n+1}^{i-1}) > 0 \\ P_{\text{LLI}}(w_i|w_{i-n+2}^{i-1}) & \text{otherwise} \ , \end{cases}$$

where $P(\cdot)$ are the probability estimates to be combined and the normalisation factor depends on the interpolation weights corresponding to each of the probability estimates. In case the context was not found in the training data, the probability estimates are taken from the lower order model, for which there is its own normalisation factor and interpolation weights. The log-linear interpolation estimate for a unigram is taken to be just a single unigram probability estimate, *i.e.*

$$P_{\mathrm{LLI}}(w_i) = P(w_i)$$

since there is no need for normalisation factor.

4.2.2 Parameter Tying

As noted above, the clustering scheme for estimating the log-linear interpolation parameters is similar to the corresponding linear interpolation scheme described in Sect.4.1.1. Since the interpolation weight is taken to only depend on a particular history and all the observed histories are clustered into a manageable number of bins, there is a separate interpolation parameter for each bin. The difference between the clustering methods, however, is in the way the interpolation weights are used. Let the mapping between the history $h = w_{i-n+1}^{i-1}$ of an n-gram w_{i-n+1}^{i} and a bin $\mathcal{B}_k$ be via the number of times this history has appeared in the kept part of the training corpus, similarly to linear smoothing,

$$\forall h \in \mathcal{K}_{\mathcal{T}} : \ \text{if} \ \ N(h) \in \mathcal{B}_k \Rightarrow \mathcal{C}_h = \mathcal{C}(\mathcal{B}_k), \ \ \mathcal{B}_k \in \mathfrak{B} \ \ ,$$

where $\mathfrak{B}$ is the collection of all possible bins, $\mathcal{B}_k$ is a certain bin into which the history count $N(h)$ falls and $\mathcal{C}_h$ is defined as a collection of parameters associated with a bin $\mathcal{B}_k$. For the linear case, the collection $\mathcal{C}_h$ consists of only one interpolation weight $\lambda(\mathcal{B}_k)$. In the log-linear framework however, given a certain n-gram history w_{i-n+1}^{i-1} of length $n-1$ there are n log-linear interpolation weights associated with it, *i.e.*

$$\forall h \ \ \Lambda(h) = \begin{cases} \lambda_1(h), \ldots, \lambda_n(h) & N(h) > 0 \\ \lambda_1(\widehat{h}), \ldots, \lambda_k(\widehat{h}) & N(h) = 0, \ \ N(\widehat{h}) > 0, \ \ k < n \end{cases} \ \ ,$$

where $\widehat{h}$ is a lower-order history. In addition, there is a normalised parameter associated with each history which is strictly dependent on both the history and the interpolation parameters associated with the history, *i.e.*,

$$Z = Z_h(\Lambda(\mathcal{B}_k)) \ \ .$$

Therefore the log-linear parameter collection for a certain $\mathcal{B}_k$ can be defined as follows

$$\forall h \in \mathcal{K}_{\mathcal{T}} : \ \text{if} \ \ N(h) \in \mathcal{B}_k \Rightarrow \mathcal{C}_h = \mathcal{C}(\mathcal{B}_k) = \{\Lambda(\mathcal{B}_k), Z_h(\Lambda(\mathcal{B}_k))\}, \ \ \mathcal{B}_k \in \mathfrak{B} \ \ ,$$

where the normalisation factors are strictly history dependent with the number of normalisation factors equal to the number of distinct histories in kept part of the training corpus.

The process of building the clusters used to estimate Λ is carried out using the *wall of bricks* clustering, described in Sect.4.1.1, and in this respect is similar to the process used for linear interpolation.

4.2.3 Multidimensional optimisation

Function to be optimised was chosen to be the log-likelihood of log-linear probability distribution which is expressed as follows

$$\mathcal{L}(\mathcal{B}_{n,k}) = \sum_{h:C(h)\in\mathcal{B}_{n,k}} \sum_{w} C(h,w) \left[-\log Z_h(\Lambda) + \sum_{i=1}^{n} \lambda_i(h) \log P(w|\widehat{h}_i) \right] \quad , \quad (4.11)$$

where $\Lambda(\mathcal{B}_k) = \lambda_1(\mathcal{B}_k), \ldots, \lambda_n(\mathcal{B}_k)$, $\widehat{h}_i$ are histories of different specificity[6] and $C(\cdot)$ denote the counts obtained from the held-out set. Conventional held-out estimation is performed using the divided training data, with the kept set used for estimating the probabilities and the held-out set used for the interpolation weights optimisation. The optimisation framework as defined by (4.11) is more complicated than that of linear interpolation one due to the presence of normalisation score, which ensures that the log-likelihood being optimised is a log-likelihood of a proper probability distribution, expressed as

$$\log Z_h(\Lambda) = \log \sum_{w\in\mathcal{K}_\mathcal{T}} P_{\mathrm{LLI}}(w|h) \quad , \quad (4.12)$$

where the summation is over all the words in the vocabulary of the kept part of the training corpus.

Because of the non-linear nature of the expression in (4.11), no closed-form solution for the first derivative exists. Unlike the linear smoothing, no fast root-finding algorithm exploiting the functional form of the log-linear log-likelihood function can be employed. However, since the expression in (4.11) describes as convex function and the second derivative may be shown to be negative, any hill-climbing unconstrained optimisation can be used. In the experiments described below, two multidimensional optimisation algorithms from a family of direct search methods not requiring derivatives were used. The first, also reported to be used by Klakow [39], is a Nelder-Mead multidimensional minimisation, also known as *simplex*, which is probably the most widely used method for nonlinear unconstrained optimisation[7], maintains at each step a non-degenerative simplex, a geometrical figure in n dimensions of nonzero volume that is the convex hull of $n+1$ vertices. It is a very computationally attractive method since it typically requires only one or two function evaluations to construct a new simplex. The rigorous theoretical analysis treating the Nelder-Mead method has recently been published by Lagarias *et al.* [42]. Despite its attractiveness, however, there are certain families of strictly convex functions for which Nelder-Mead method converges to a non-stationary point, as shown by McKinnon [53], who considered certain functions of two variables. Some remedies for detection of non-optimality were proposed by Kelley [38]. Due to this recent evidence of potential problems with Nelder-Mead method, it has been applied in experiments on the bigger, less sparse corpora, whereas for the smaller corpora, a generally slower, but more robust Powell method was employed, a direction set method whose choice of successive directions does not require the calculation of a gradient and which employs Brent one-dimensional search in each direction [62]. It is not clear, however, whether the log-likelihood family of functions can produce functions which may cause problems for unconstrained direct search optimisation.

[6] With a zero length history corresponding to a unigram.

[7] Not to be confused with Dantzig's simplex algorithm for linear programming.

No convergence problems whatsoever with both of the techniques were observed during the experiments.

The log-likelihood optimisation is carried out in top-down fashion, in case of trigram smoothing starting with the trigrams and ending with the bigrams, as shown in Alg.4.3. There are several important points which are worth mentioning. Firstly, unlike linear smoothing, where all the histories from the held-out set (except those, obviously, which are not found in the kept set) are used in optimisation, because there is only one weight corresponding to a certain history of an n-gram of length m, whereas in case of log-linear smoothing, for a certain n-gram history of length m there are m weights associated with it, namely one for each of the m sub-histories of an n-gram. Therefore, when optimising the trigrams, for instance, not only a specific weight for a trigram is optimised, but also the weights for each of the lower-order n-gram models (namely bigram and unigram). While optimising bigrams, however, it is desirable to exclude all the bigram histories with corresponding weights which were already optimised during the trigram optimisation stage in order to avoid duplication of the data for the log-likelihood optimisation. Hence, for the trigram log-linear interpolation model, following information is included in optimisation:

trigram optimisation is carried out for all the trigram histories (and, consequently, for the corresponding lower-order bigram histories) in the held-out set which were also found in the kept set.

bigram optimisation is done for all the bigram histories common to the kept and the held-out set which are subset of the held-out set trigrams that were not found in the kept set.

This way of collecting the histories ensures that the sets of data used for trigram and bigram-level optimisation are disjoint.

In addition, as can be seen from equation (4.11), in the actual optimisation process, the histories belonging to a kept set which are not found in the held-out set, do not influence the optimisation process (because their associated counts $C(\cdot)$ are zero). Therefore, during the actual optimisation, the normalisation factors given by (4.12) are only calculated for those histories that are common to both the held-out and the kept set. After the optimisation for a certain bin is finished, the normalisation factors are calculated for the rest of the histories belonging to a bin using the optimal weights. This dramatically reduces the computational complexity of that part of the algorithm which recalculates the normalisation factors for the histories in a certain bin.

The sole computational bottleneck of the log-linear algorithm, which makes it orders of magnitude slower than linear smoothing, is the calculation of normalisation scores over the whole vocabulary. An interesting alternative is suggested by Chen *et al.* [16] and is applicable to all forms of exponential language models requiring normalisation. The proposed method is to ignore the normalisation factors completely and consider scores (with special processing required to prevent scores from rising above 1) instead of probabilities, which makes the model as fast as conventional language models and can easily be used to calculate the Word Error Rate (WER) in expensive speech recognition tasks such as lattice rescoring. The major disadvantage of the suggested technique is that it is not possible anymore to calculate the perplexity of the model.

for all n-gram history orders m starting with n-grams ($m = n - 1$) **do**
  Collect and cluster the histories h from the held-out set $\mathcal{H}_\mathcal{T}$ corresponding to n-grams history length m:
  for all histories $\{h\}$ of length m from the held-out set $\mathcal{H}_\mathcal{T}$ **do**
    if h is a subset of a higher order history already observed during the previous iteration of optimisation **then**
      Do not include such history h in optimisation.
    else
      Include h in optimisation.
    end if
  end for
  Perform the per-bin optimisation:
  for all bins $\mathcal{B}_{n,k}$ in a collection $\mathfrak{B}_n$ **do**
    Given the current bin $\mathcal{B}_{n,k}$, run optimisation where:
    for each guess of interpolation weights $\Lambda(\mathcal{B}_{n,k})$ **do**
      for all histories h from $\mathcal{H}_\mathcal{T}$ which were not previously excluded from optimisation and which fall into a bin $\mathcal{B}_{n,k}$ **do**
        Calculate normalisation scores $\log Z_h(\Lambda(\mathcal{B}_{n,k}))$ given by (4.12) over the kept set $\mathcal{K}_\mathcal{T}$ vocabulary.
      end for
      Obtain the next guess $\Lambda'(\mathcal{B}_{n,k})$ maximising the log-likelihood $\mathcal{L}(\Lambda'(\mathcal{B}_{n,k}))$ defined by (4.11).
    end for
    Given the final estimates of $\Lambda(\mathcal{B}_{n,k})$, calculate normalisation factors for *all* the histories h from the kept set $\mathcal{K}_\mathcal{T}$ belonging to a bin $\mathcal{B}_{n,k}$.
  end for
end for

Algorithm 4.3: Top-down log-likelihood optimisation for an n-gram log-linear interpolation model.

4.2.4 Similarity to Maximum Entropy models

The goal of maximum entropy language modelling is to construct a model of the process that generated the training data $\tilde{P}(h, w)$ [4]. The expected value of binary indicator function (also known as *feature function*) f ($f \in \{0, 1\}$) with respect to the empirical distribution $\tilde{P}(h, w)$ is

$$\tilde{P}(f) \triangleq \sum_{h,w} \tilde{P}(h, w) f(h, w) \ .$$

The baseline maximum entropy models usually use nested trigram, bigram and unigram features with $(h, w) = (u, v, w)$ where the trigram feature, for instance, is defined as

$$f_{uvw}(\tilde{u}, \tilde{v}, \tilde{w}) = \begin{cases} 1 & \text{if } w = \tilde{w} \text{ and } v = \tilde{v} \text{ and } u = \tilde{u} \\ 0 & \text{otherwise .} \end{cases}$$

When the statistic which is considered to be "useful" is discovered, its importance is acknowledged by requiring the model to accord with it by constraining the expected value that the model assigns to the corresponding feature function f. The expected

value of f with respect to the model $P(w|h)$ is

$$P(f) \triangleq \sum_{h,w} \tilde{P}(h)P(w|h)f(h,w) \ ,$$

where $\tilde{P}(h)$ is the empirical distribution of histories h in the training data. This expected value is then constrained to be the same as the expected value of f in the training data by requiring

$$P(f) = \tilde{P}(f) \ .$$

Combining the above three equations yields

$$\sum_{h,w} \tilde{P}(h)P(w|h)f(h,w) = \sum_{h,w} \tilde{P}(h,w)f(h,w) \ ,$$

which is a *constraint* that excludes from consideration all those models which do not agree with the training data on how often the output of a process should exhibit the feature f. Given the n feature functions f_i which determine the statistics one feels are important in modelling the process, the idea is to make the model accord with these statistics. The conditional entropy of the distribution $P(w|h)$ is

$$H(P(w|h)) = -\sum_{h,w} \tilde{P}(h)P(w|h)\log P(w|h) \tag{4.13}$$

and according to the principle of the maximum entropy the model maximising the entropy in (4.13) should be selected from a set $\mathcal{C}$ of allowed probability distributions. The problem is treated as constrained optimisation problem, with each feature f_i being assigned a Lagrangian multiplier γ_i, with the optimisation function given by

$$\mathcal{L}(P(w|h),\Gamma) \triangleq H(P(w|h)) + \sum_i \gamma_i(P(f_i) - \tilde{P}(f_i)) \ . \tag{4.14}$$

Holding $\Gamma = \{\gamma_i\}$ fixed, the unconstrained maximum of (4.14) is given by

$$P_\Gamma(w|h) = \frac{1}{Z_\Gamma(h)} \exp\left(\sum_i \gamma_i f_i(h,w)\right), \tag{4.15}$$

where the normalisation is defined as

$$Z_\Gamma(h) = \sum_w \exp\left(\sum_i \gamma_i f_i(h,w)\right)$$

and the value of $\mathcal{L}(P(w|h),\Gamma)$, also known as *dual* function, at maximum can be shown to be

$$\Psi(\Gamma) = -\sum_h \tilde{P}(h)\log Z_\Gamma(h) + \sum_i \gamma_i \tilde{P}(f_i) \ .$$

The maximum entropy language model can thus be defined as a model which is subject to the constraints $\mathcal{C}$, has the parametric form of (4.15), with the optimal parameters Γ_{opt} determined by maximising the dual function $\Psi(\Gamma)$, *i.e.*

$$\Gamma_{\text{opt}} = \arg\max_\Gamma \Psi(\Gamma) \ .$$

As can be seen, there are some obvious parametric similarities between the log-linear probability estimate given by (4.10) and the maximum entropy equation (4.15). In addition, both techniques employ the same functional form of normalisation.

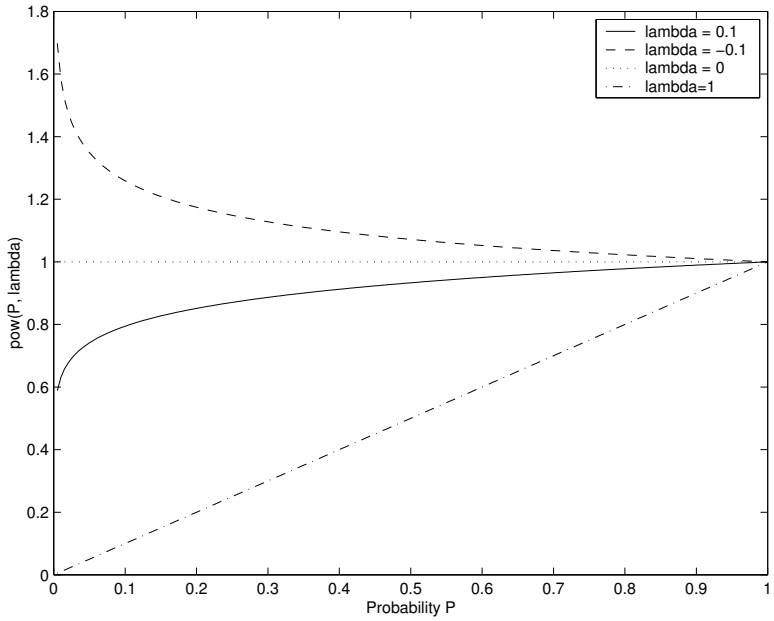
4.2.5 Significance of log-linear interpolation weights

As it was noted above, the log-linear interpolation weights can have any value since they are in essence the redefined Lagrangian multipliers. It is interesting to check how the log-linear interpolation weights influence the exponential term in the product (4.10). According to the different possible ranges into which λ might fall, there are several possible scenarios, shown in Table 4.1. In this particular context, there are two possible cases. The first case shown in Figure 4.1(A), when the absolute value of λ is less than one in which case the values of λ used in the graph are 0.1 and -0.1. In such a case, for strictly positive weights the contribution of the small probabilities is strongly attenuated and for the negative weights the small probabilities are strongly boosted. In the asymptotic case, which occurs for higher probabilities, their overall contribution tends to one. The simplest case is when the log-linear interpolation weight is zero, which essentially means that the term in question becomes equal to one, which is equivalent to ignoring this particular probability in the product given by equation (4.10). In the case when the weight is equal to one, the probability estimate is accurate enough and is neither boosted nor attenuated.

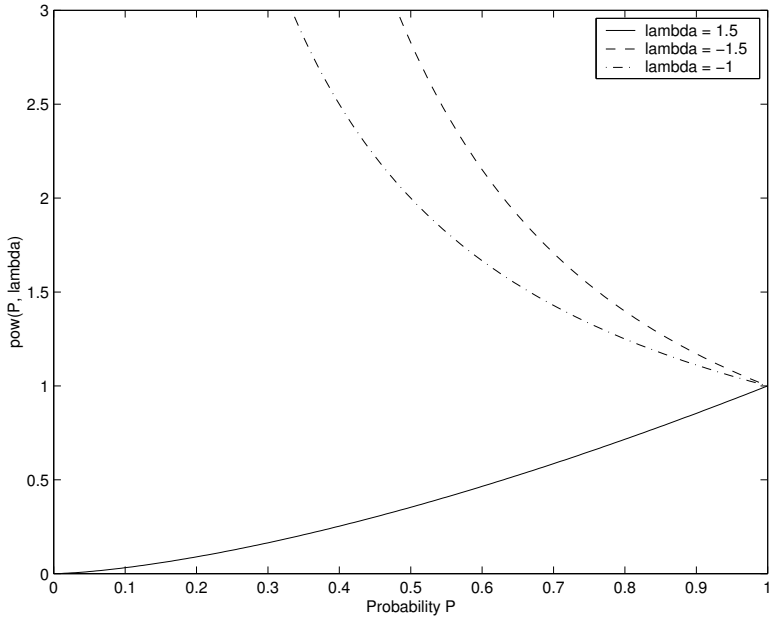
The second interesting case, shown in Figure 4.1(B), occurs when the absolute value of λ is bigger than one or equal to minus one. In case of a negative weight $\lambda \leq -1$, the small probabilities are very strongly boosted with weak boosting for bigger probabilities. However, the boosting is still stronger than in the first case, even for the big probabilities. For positive weight, as can be seen from the graph, such a weight effectively cancels the small probabilities, with the overall contribution of such probabilities becoming very small, while boosting the bigger probabilities.

In order to investigate the behaviour of the expression P^{λ} and its related counterpart in the log-linear domain, namely $\lambda \log P$, a function of the two variables P and λ was investigated. The overall behaviour of the function P^{λ} and its logarithmic counterpart $\lambda \log P$ as a function of probability and log-linear interpolation weights is summarised in Figure 4.2. It should be noted that the singularities of the function has been avoided by ensuring that no probability P and weight λ can be both equal to zero at the same time. The boosting and attenuation effects of different weight and probability ranges are evident on both of the graphs.

At this stage it can be concluded that the log-linear interpolation weights have different interpretation from that of the regular linear interpolation weights. By not constraining the log-linear interpolation weights to be in the certain range, there are more possible cases to consider when analysing the specific log-linear interpolation model.

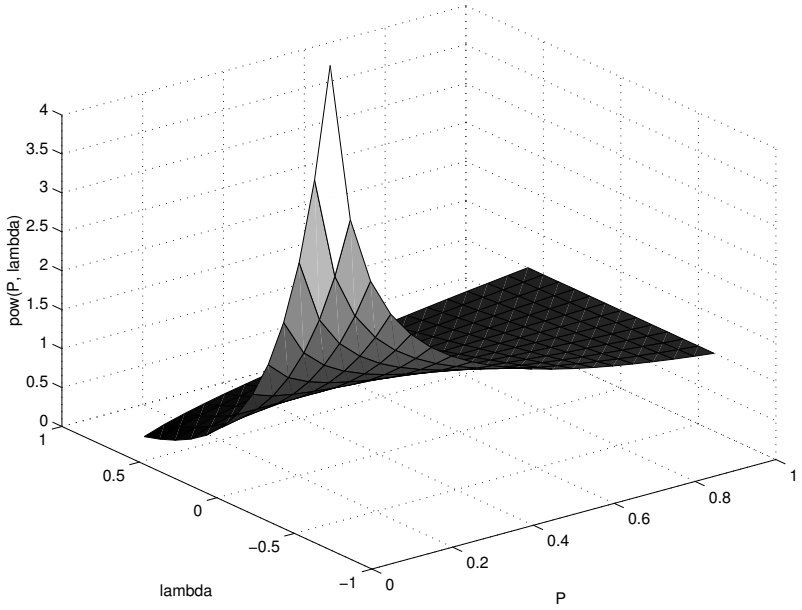


(a) P^{λ}: $|\lambda| \leq 1$

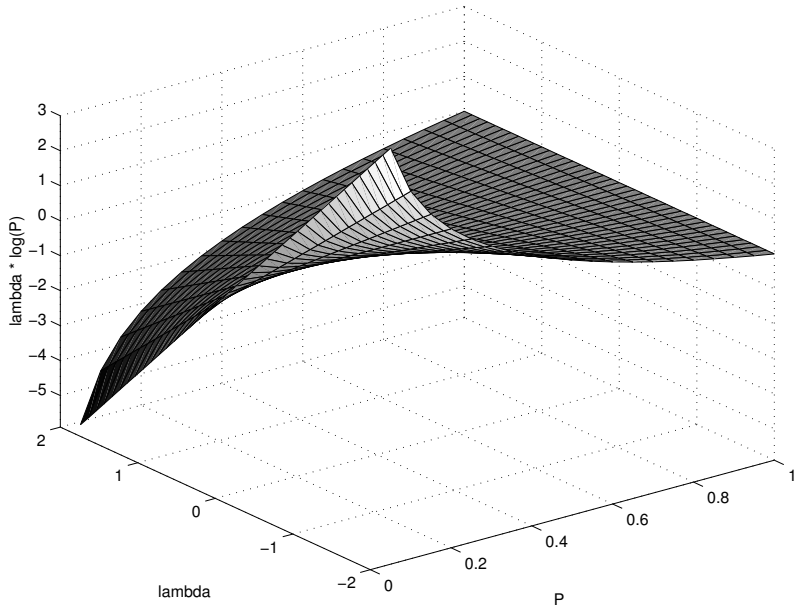


(b) P^{λ}: $|\lambda| > 1$

Figure 4.1: Plot of the unnormalised log-linear interpolation term as function of probability P with the weight values λ being taken from different ranges.



(a) P^{λ}



(b) $\lambda \log P$

Figure 4.2: Unnormalised log-linear interpolation term surface as a function of both the probability P and the log-linear weight λ.

CHAPTER 5

Performance Evaluation

This chapter presents some of the experimental results obtained for linear and log-linear interpolation of trigram models. Section 5.1 describes the text corpora used in the experiments, section 5.2 describes the baseline back-off models with which the resulting linear and log-linear interpolation models are compared, section 5.3 describes the experiments with the trigram linear interpolation model and compares it with the performance of trigram Katz back-off model and section 5.4 describes the results obtained for log-linear interpolation experiments and compares them with the results for linear interpolation and back-off models.

5.1 Language Modelling Corpora

In this section the language modelling corpora used for training and testing the language models are described. The two text corpora selected from the wide range of the corpora available for language modelling, are the transcriptions of conversational telephone speech, also known as Switchboard, and the archive of Wall Street Journal articles. The first corpus is a good sample of real conversational speech, whereas the second is a wide coverage representation of a newswire text. It can also be noted that the sizes of the corpora are different, with the Switchboard data being smaller, which is important for testing the behaviour of the language models on corpora of different sizes.

5.1.1 Transcriptions of Conversational Telephone Speech

Current experiments for conversational telephone speech are often conducted on three corpora distributed by the Linguistic Data Consortium (LDC): Switchboard-I, Switchboard-II and Callhome English. Both Switchboard corpora consist of telephone conversations within the USA between strangers. These corpora are the subject of the yearly Hub5 evaluation for large vocabulary continuous speech recognition (Hub5-LVCSR) conducted by the National Institute for Standards and Technology (NIST).

The Switchboard Telephone Speech Corpus was originally collected by Texas Instruments in 1990-1, under DARPA sponsorship. The first release of the corpus was published by NIST and distributed by the LDC in 1992-3 [26]. Since that release, a number of corrections have been made to the data files. Switchboard-I is a collection of about 2400 two-sided telephone conversations among 543 speakers (302 male, 241 female) from all areas of the United States. A computer-driven "robot

Table 5.1: Hub5 language modelling corpus details (sizes of the corpora are shown in total number of words).

Set	Size (words)	Vocabulary size
`Hub5-Train00`	3,636,489	27,754
`Hub5-Eval97`	48,456	3,262
`Hub5-Eval98`	47,084	3,003

operator" system handled the calls, giving the caller appropriate recorded prompts, selecting and dialling another person (the callee) to take part in a conversation, introducing a topic for discussion and recording the speech from the two subjects into separate channels until the conversation was finished. About 70 topics were provided, of which about 50 were used frequently. Selection of topics and callees was constrained so that no two speakers would converse together more than once and no one spoke more than once on a given topic.

Switchboard-II consists of 3,638 5-minute telephone conversations involving 657 participants. This corpus was collected by the Linguistic Data Consortium (LDC), in support of a project on Speaker Recognition sponsored by the U.S. Department of Defence. Each recruit was asked to participate in at least 10, 5-minute phone calls. Ideally each participant would receive 5 calls at a designated number and make 5 calls from phones with different telephone numbers (ANI codes). A suggested topic of discussion was given (read by the automated operator), although participants could chat about whatever they preferred.

From both of the aforementioned Hub5 Switchboard transcriptions, approximately 3.6 million words were available for language model training, and the two transcriptions of evaluation sets 1997 Hub5 and 1998 Hub5 were used for testing. Table 5.1 shows the details for the training portion of the Hub5 language modelling corpus, denoted by `Hub5-Train00`, and for the two evaluation sets, denoted by `Hub5-Eval97` and `Hub5-Eval98`.

5.1.2 The Wall Street Journal (WSJ) Corpus

The WSJ corpus contains newspaper text collected from the Wall Street Journal over the period 1987-1989 inclusive [60]. Since this corpus is considerably bigger than the Hub5 Switchboard language modelling corpus, findings made using it are expected to hold also for larger corpora such as North American Broadcast News (NAB), on which many state-of-the-art speech recognition systems have been based [72].

The portion of the corpus selected for training consists of 1988 Wall Street Journal archive denoted as `WSJ-Train88` and the three evaluation sets from years 1987 to 1989, denoted as `WSJ-Eval87`, `WSJ-Eval88` and `WSJ-Eval89`, with the details the training and evaluation sets shown in Table 5.2. In the following experiments with the WSJ corpora following settings were used:

- Standard vocabulary of 65,464 words (as used in [73], where significant reduction in the OOV rate was reported in comparison with the CMU vocabulary of 20,000 words) from North American Broadcast News (NAB) corpus used for training the models.

Table 5.2: WSJ language modelling corpus details (sizes of the corpora are shown in total number of words).

Set	Size (words)	Vocabulary size
WSJ-Train88	17,057,940	107,065
WSJ-Eval87	22,452	3,793
WSJ-Eval88	21,947	4,092
WSJ-Eval89	22,489	4,036

Table 5.3: Total number of words in the kept and held-out portions of the training data for Hub5-Train00 and WSJ-Train88 together with the wordlist sizes.

Set	Kept Set $\lvert\mathcal{K}_\mathcal{T}\rvert$	Held-out Set $\lvert\mathcal{H}_\mathcal{T}\rvert$	Vocabulary $\lvert\mathcal{V}\rvert$
Hub5-Train00	3,527,238	109,251	27k
WSJ-Train88	16,989,671	68,240	65k

- All the singleton events were discarded.
- All the evaluation sets were obtained from the WSJ language model development data for the corresponding year and constrained to have approximately 20,000 words.
- Symbol marking the beginning of a sentence was not predicted during the language model testing.

5.2 Baseline Models

The division of the training data $\mathcal{T}$ necessary for using the cross-validation, was done for both the 2000 Hub5 language modelling training data (Hub5-Train00) and the 1988 WSJ training data (WSJ-Train88), with the sizes of the kept parts $\mathcal{K}_\mathcal{T}$ and the held-out parts $\mathcal{H}_\mathcal{T}$ shown in Table 5.3 together with the corresponding word list sizes $|\mathcal{V}|$ used in the experiments. As can be seen from the table, the held-out portion of a larger WSJ training corpus is smaller than the held-out portion of the Switchboard language model training corpus. There was no particular reason for such a division, with the only consideration being that the held-out set should be significantly smaller than the kept set.

The baseline model for the Hub5 evaluation was chosen to be the trigram back-off model using the Good-Turing discounting with frequency of frequencies (FoF) discounting range of 8 and no event or context cutoffs. The baseline model for the WSJ evaluations was chosen to be the trigram back-off model with FoF range of 8 and with the singleton events cutoffs applied to trigrams and bigrams. Both back-off models were built on the whole training corpus $\mathcal{T}$ to ensure that back-off models and linear and log-linear interpolation models use the same overall amount of training data. Perplexities of the back-off models with respect to the evaluation sets described in previous section are shown in Table 5.4.

Table 5.4: Perplexities of the baseline back-off models trained on 2000 Hub5 language model training data and on 1988 WSJ training data, tested on the respective evaluation sets.

Model	Test Set	Perplexity
Hub5-Train00	Hub5-Eval97	98.2
	Hub5-Eval98	97.5
WSJ-Train88	WSJ-Eval87	165.2
	WSJ-Eval88	199.3
	WSJ-Eval89	198.0

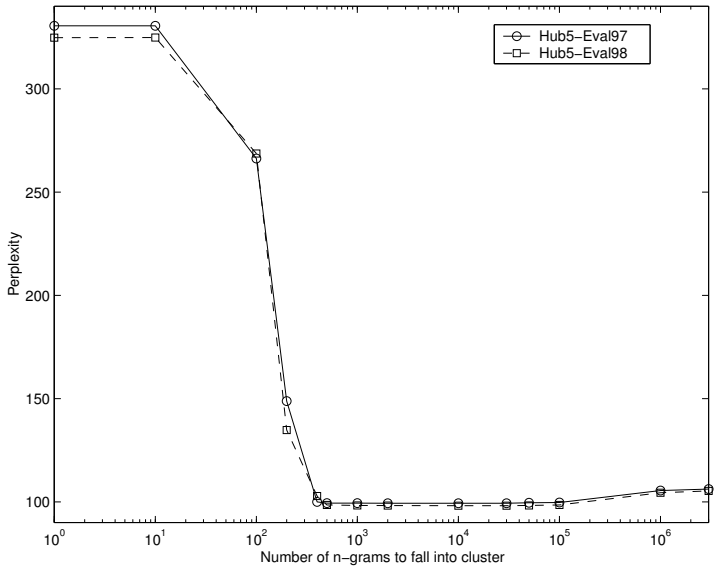


Figure 5.1: Influence of cluster size $N_{\min}$ on the performance of linear interpolation of maximum likelihood estimates.

5.3 Linear Interpolation

In this section the experiments with trigram linear interpolation models carried over Hub5 Switchboard and Wall Street Journal language modelling corpora are described. The results are presented for two different cases, the first one concerning the linear smoothing of the maximum likelihood estimates and the second one the linear smoothing of Katz back-off models.

Maximum likelihood estimates

First, the optimal clustering scheme for linear interpolation of maximum likelihood estimates, was selected by experimenting with different values of clustering constraint $N_{\min}$ defined in Sect.4.1.1, which essentially defines the bins for the interpolation weights. Influence of this parameter on the overall performance of the linear interpolation model on two of the abovementioned data sets is shown in Figure 5.1. The worst result is obtained when no clustering is used at all, corresponding to $N_{\min} = 1$, by keeping a separate interpolation weight for each n-gram history. In

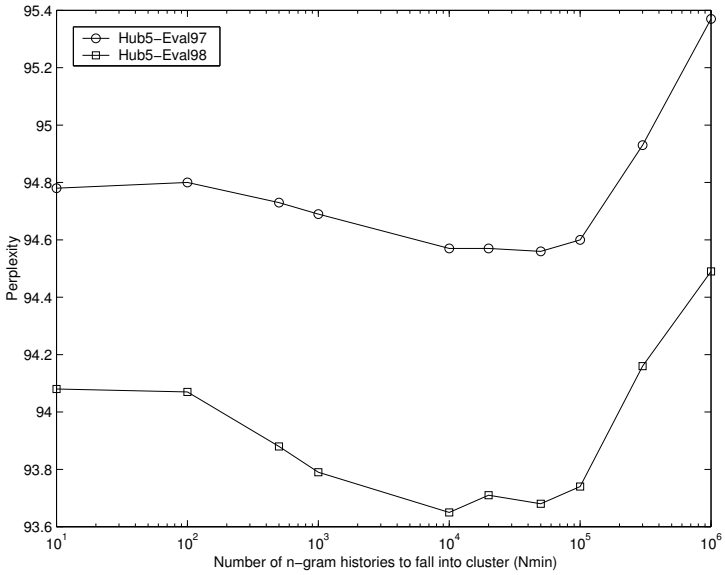


Figure 5.2: Influence of cluster size $N_{\min}$ on the performance of linear interpolation of Katz back-off models.

this case there is never going to be enough data to accurately estimate such an enormous number of parameters and the resulting perplexities are high. The optimal perplexities, obtained on both test sets and found to correspond to $N_{\min} = 10^4$, are shown in Table 5.5, with perplexity on the `Hub5-Eval97` test set equal to 99.3 and the perplexity on the `Hub5-Eval98` test set equal to 98.1, both slightly higher than the corresponding baseline back-off model perplexities.

Back-off estimates

According to the expectations, the linear interpolation of the back-off probability estimates should perform at least as well as the back-off model itself. In order to check this claim, recursive model consisting of linear combination of back-off scores, where the recursion is terminated by taking the back-off probability estimate of a unigram, defined similarly to (4.3) as

$$\begin{aligned} P_{\mathrm{LI}}(w_i|w_{i-n+1}^{i-1}) &= (1-\lambda_{w_{i-n+1}^{i-1}})P_{\mathrm{BO}}(w_i|w_{i-n+1}^{i-1}) + \\ &\quad \lambda_{w_{i-n+1}^{i-1}} P_{\mathrm{LI}}(w_i|w_{i-n+2}^{i-1}) \end{aligned} \tag{5.1}$$

was built and tested. Each back-off estimate was obtained from the conventional Katz smoothing model, introduced in Sect.3.2.1[1], with the basic Good-Turing discounting applied to small counts. Similarly to the maximum likelihood case discussed above, the probability estimates were obtained from the kept part of the training set and the interpolation parameters were trained on the held-out set. Figure 5.2 shows the results of the optimal clustering experiment conducted on the two Hub5 test sets. By comparing the performance of the conventional back-off model trained using the whole training corpus (both held-out and kept parts) with the linear combination of back-off estimators presented above, the superiority of the latter

[1]The discounting frequency used was $k = 8$ and no context or event cut-offs were applied.

Table 5.5: Performance of the linear interpolation models (maximum likelihood and back-off estimates) with clustering constraint of $N_{\min} = 10^4$ and back-off language model on the two test sets from Hub5.

Models	Perplexity	
	`Hub5-Eval97`	`Hub5-Eval98`
back-off	98.2	97.5
linear interp. (ML)	99.3	98.1
linear interp. (back-off)	**94.5**	**93.6**

model is evident regardless of the clustering criteria. The optimal clustering parameter $N_{\min}$, for which the biggest perplexity reduction was achieved, was yet again found to be $N_{\min} \approx 10^4$. The value of optimal clustering parameter depends on the size of the training set and on the ratio δ between the held-out set and the kept set sizes. It may be expected that the value of the optimal parameter corresponding to the different layout of the training data will be different.

The optimal perplexities of both best models, corresponding to clustering criterion of $N_{\min} = 10^4$, on two test sets are shown in Table 5.5. As can be seen, although the back-off model outperforms the linear interpolation of maximum likelihood estimates on the both test sets, the differences in perplexities are small. Linear interpolation of back-off estimates, however, consistently outperforms the back-off model and the reduction of perplexity achieved over the baseline back-off model is 3.7% on the `Hub5-Eval97` test corpus and 4% on `Hub5-Eval98` test corpus.

5.4 Log-Linear Interpolation

In the following experiments, the probability estimates to be smoothed are taken to be conventional Katz back-off estimates using Good-Turing discounting. The experiments were conducted on both Hub5 Switchboard and WSJ corpora.

5.4.1 Performance on Hub5 1997/1998 evaluation sets

Trigram log-linear interpolation model was built using the 2000 Hub5 language modelling data `Hub5-Train00`, which was divided into the kept and held-out portions, as described in Sect.5.2, where similarly to linear smoothing, kept part of the corpus was used for obtaining the trigram, bigram and unigram Katz back-off probability estimates, and the held-out portion was used for optimising the log-linear smoothing weights. Powell multidimensional direct search [62] was used for the parameter optimisation.

The influence of the clustering constraint $N_{\min}$ on the performance of the model on the two test sets against the baseline linear smoothing models (described in previous section) is shown in Figure 5.3. As can be noted from the figure, the log-linear smoothing is better behaved for the high values of clustering parameters $N_{\min}$ with the smaller rate of increase in perplexity than in the linear smoothing case. For smaller values of clustering parameters, however, linear smoothing performs in a more stable manner. Both linear and log-linear smoothing models outperform the baseline back-off models by 3.5% to 4%, with the log-linear interpolation model performing slightly worse than its linear counterpart on the `Hub5-Eval98` test set

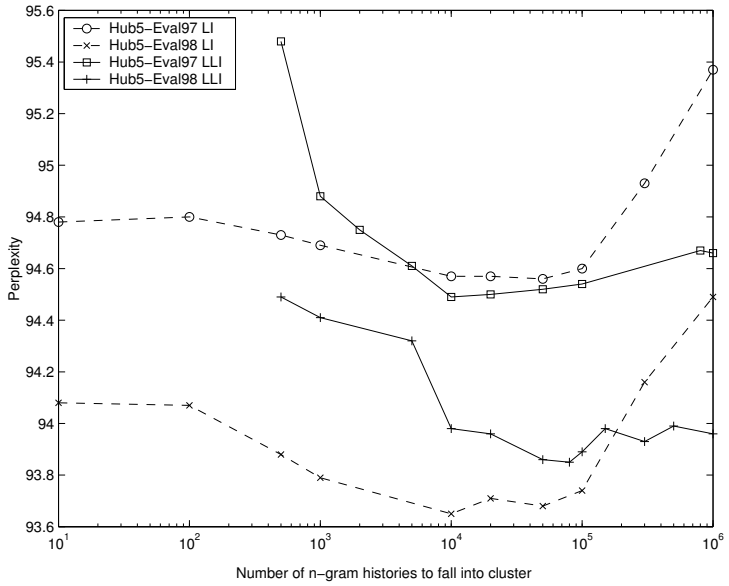


Figure 5.3: Influence of cluster size $N_{\min}$ on the performance of log-linear interpolation of Katz back-off estimates on the Hub5 1997/1998 language modelling evaluation sets.

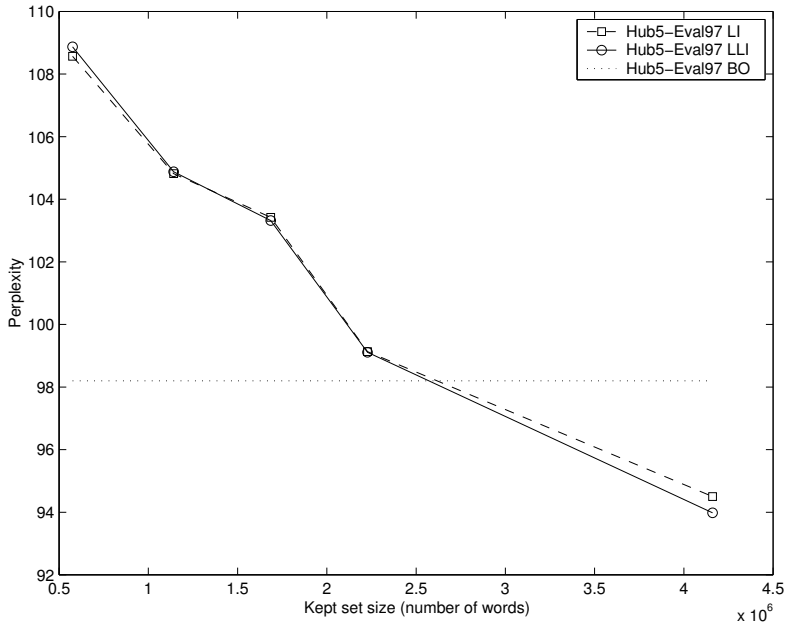
Table 5.6: Perplexity of log-linear smoothing model built on 2000 Hub5 language modelling data using $N_{\min} = 10^4$ versus its linear counterpart.

Models	Perplexity	
	`Hub5-Eval97`	`Hub5-Eval98`
linear (back-off)	94.6	93.7
log-linear (back-off)	**94.5**	**93.9**

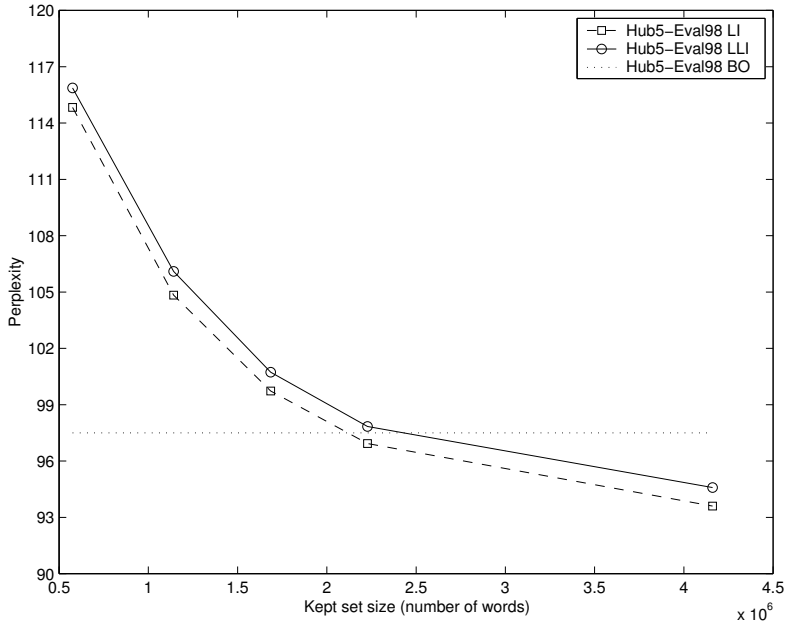
and slightly better on the `Hub5-Eval97` test set. The perplexities of the best log-linear interpolation model selected among all of the interpolation models built on the 2000 Hub5 language modelling corpus and the corresponding best linear interpolation baseline model, both interpolating Katz back-off estimates, given the best clustering parameter $N_{\min} = 10^4$ are summarised in Table 5.6. As can be seen from the table, the reduction in perplexity of the best log-linear model on the `Hub5-Eval97` test set with respect to the best linear model is tiny.

The next experiment investigated the influence of the size of the kept portion of 2000 Hub5 training data on the performance of the log-linear interpolation model, which involved changing of the kept set size while keeping the held-out set size fixed. The performance of the resulting linear and log-linear interpolated language models with respect to baseline back-off model is shown in Figure 5.4. It can be noted that increasing the size of the kept set in depicted range of the corpus sizes improves the performance of the models with respect to the baseline back-off model which was trained on the maximum amount of training data available.

It is interesting to compare the optimal weights obtained both linear and log-linear smoothing methods. Optimal weights for the the case when all the histories fall into one cluster are shown in Table 5.7. Because of the recursive definition of the linear smoothing framework of Sect.4.1, the weights of the lower-order model



(a) `Hub5-Eval97`: Baseline back-off perplexity is 98.2



(b) `Hub5-Eval98`: Baseline back-off perplexity is 97.5

Figure 5.4: Influence of the size of the kept portion of 2000 Hub5 training set on the performance of linearly and log-linearly smoothed Katz back-off estimates.

Table 5.7: Optimal linear and log-linear interpolation weights trained on 2000 Hub5 language model training corpus with $N_{\min} = 10^6$, with the pair of interpolation weights corresponding to bigram model, and the 3-tuple to the trigram model.

Smoothing	Weights				
	bigram		trigram		
	λ_0^{bi}	λ_1^{bi}	λ_0^{tri}	λ_1^{tri}	λ_2^{tri}
linear	0.012123	0.987877	0.0017154	0.139786	0.858498
log-linear	−0.014994	0.967262	0.0023987	0.037057	0.896499

are reused in a higher-order model. In trigram case, for instance,

$$\begin{aligned} \lambda_2^{tri} &= 1 - \lambda^{tri} \\ \lambda_1^{tri} &= \lambda^{tri}\lambda_1^{bi} \\ \lambda_0^{tri} &= \lambda^{tri}(1 - \lambda_1^{bi}) \end{aligned}$$

and, for bigrams

$$\begin{aligned} \lambda_1^{bi} &= 1 - \lambda^{bi} \\ \lambda_0^{bi} &= \lambda^{bi} \ , \end{aligned}$$

where the two weights λ^{bi} and λ^{tri} uniquely define the trigram linear interpolation model. In case of log-linear smoothing, unlike linear interpolation, there is no apparent relation between the weights, they do not necessarily sum to one and can be negative which can be explained by the specific way in which the log-linear framework was defined, in which all the five weights are essential for defining the model. However, by examining the table, it can be noted that the absolute value of all the log-linear weights is less than one, and although they do not necessarily sum to one, they are not very different from the corresponding linear smoothing weights. An additional interesting observation that has been made during the experiments concerns the values of the log-linear weights and is summarised in Table 5.8, where the total number of optimal weights whose absolute value is bigger than one is shown against the total number of optimal weights calculated for each level of a trigram log-linear model. An overall statistic for the Hub5 corpus shows that absolute values of approximately 21% of the total number of weights calculated exhibited the property of being more than one. In particular, the upper bound on such absolute values was found to be approximately equal to 1.3.

5.4.2 Performance on 1987–1989 WSJ evaluation sets

For the experiments with the Wall Street Journal archives, the trigram log-linear interpolation models were built using the 1988 WSJ archive `WSJ-Train88`, where the Katz back-off estimates to be smoothed were obtained from the kept portion of the training data, 65k vocabulary was used and event cutoffs were applied to singleton bigram and unigram distributions. Nelder–Mead multidimensional direct search was used for optimising the smoothing weights. In order to compare the performance of the log-linear smoothing on this corpus, the baseline linear interpolation models and Katz back-off models were built using the same settings.

Table 5.8: Total number of optimal log-linear smoothing weights calculated for each level of all the log-linear trigram language models trained during the experiments against the total number of optimal weights whose absolute values exceeded unity for Hub5 1997/1998 language model evaluation sets.

Model Level	# λ	# $\lvert\lambda\rvert > 1$
bigram	2008	456
trigram	3669	726
overall	5677	1182

Table 5.9: Optimal linear and log-linear interpolation weights trained on 1988 WSJ language model training corpus with $N_{\min} = 10^7$, with the pair of interpolation weights corresponding to bigram model, and the 3-tuple to the trigram model.

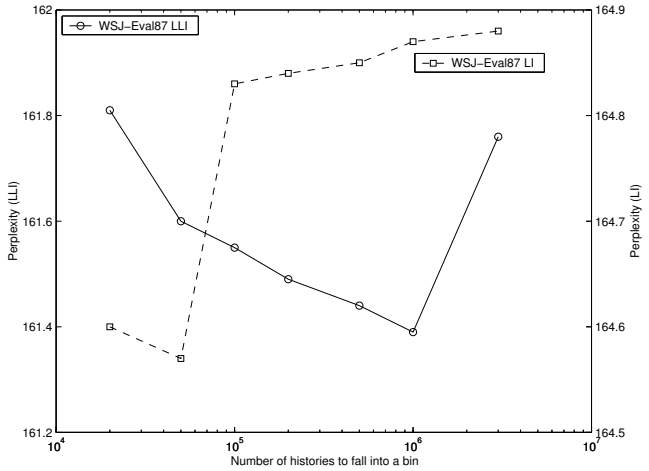
Smoothing	Weights				
	bigram		trigram		
	λ_0^{bi}	λ_1^{bi}	λ_0^{tri}	λ_1^{tri}	λ_2^{tri}
linear	0.0066299	0.993370	0.0003469	0.05198306	0.94767
log-linear	0.0017205	0.965193	0.0225268	−0.0262599	0.94396

Optimal weights obtained for the case where all the histories fall into one bin, corresponding to $N_{\min} = 10^7$, for both linear and non-linear smoothing are shown in Table 5.9. Similar to a Hub5 case, the absolute values of the vast majority of the weights obtained were less than 1.

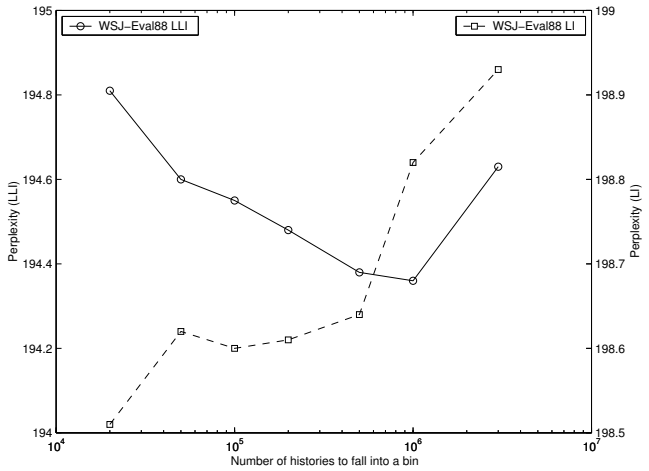
Figure 5.5 displays the performance of the trigram log-linear model on the three evaluation sets from the WSJ corpus. The performance of the corresponding baseline linear interpolation models is displayed on the same plot for convenience. Performance of the log-linear model, as compared to the baseline Katz back-off and linear interpolation, on the three test sets is summarised in Table 5.10. As can be seen, log-linear interpolation compares favourably with the baseline techniques, with the perplexity reduction of 2.3% on the `WSJ-Eval87` test set, 2.5% on the `WSJ-Eval88` test set and 3.2% on the `WSJ-Eval89` test set with respect to the Katz back-off baseline model. Moreover, log-linear interpolation outperforms linear interpolation, with the reduction in perplexity of 2% on the `WSJ-Eval89` test set, 2.1%

Table 5.10: Performance of the log-linear smoothing model built on 1988 WSJ language modelling data using versus its back-off and linear counterparts (best models, in terms of the clustering criterion) were selected).

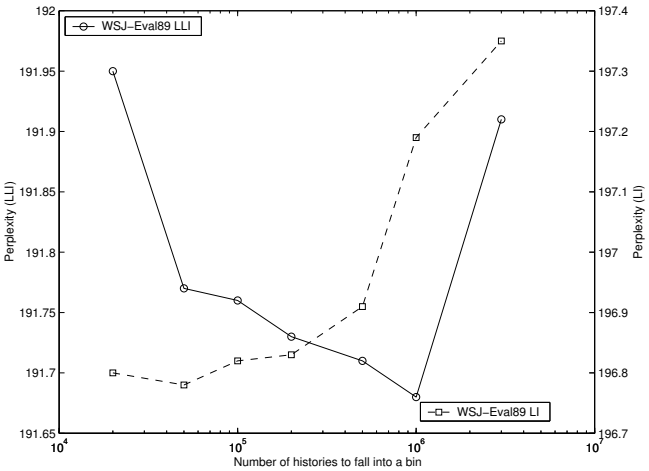
Models	Perplexity		
	`WSJ-Eval87`	`WSJ-Eval88`	`WSJ-Eval89`
back-off	165.2	199.3	198.0
linear (back-off)	164.6	198.5	196.8
log-linear (back-off)	**161.4**	**194.4**	**191.7**



(a) `WSJ-Eval87`: Baseline back-off perplexity is 165.2



(b) `WSJ-Eval88`: Baseline back-off perplexity is 199.3



(c) `WSJ-Eval89`: Baseline back-off perplexity is 198.0

Figure 5.5: Performance of trigram log-linear and linear interpolation models on 1987/88/89 WSJ evaluation sets.

Table 5.11: Value of clustering parameter $N_{\min}$ for which the optimal perplexities are obtained on the three WSJ evaluation sets for linear and log-linear interpolation models.

Models	$N_{\min}^{\text{opt}}$		
	WSJ-Eval87	WSJ-Eval88	WSJ-Eval89
linear (back-off)	50,000	20,000	20,000
log-linear (back-off)	1,000,000	1,000,000	1,000,000

on the WSJ-Eval88 test set and 2.6% on the WSJ-Eval89 test set.

From this graph it can be seen that for all the three experiments carried out on three different test corpora, the values of clustering parameters for which the optimal perplexities for linear and log-linear interpolation models are obtained are not necessarily coinciding. The values of optimal clustering parameters $N_{\min}^{\text{opt}}$ yielding optimal perplexities on the three test sets for linear and log-linear interpolation models are shown in Table 5.11. The difference in values for optimal clustering parameters can be explained by the fact that although the histories are clustered according to the same criterion, the way the histories are pooled across the held-out set is different for the two interpolation schemes and, eventually, different number of histories end up in the the bin whose characteristics, according the kept set counts, are identical for the linear and log-linear interpolation models.

5.5 Discussion

In this chapter, the performance of log-linear interpolation of language models as a smoothing technique for n-gram probability estimates has been measured through the perplexity of the test corpora, the models were trained and tested on the Hub5 Switchboard language modelling corpora and the Wall Street Journal archives. The baseline models were chosen to be the conventional Katz back-off and linear interpolation models. Whenever the performance of the log-linear interpolation model was compared to that of a particular baseline model, a care was taken not to discriminate one model against the other. When comparing with linear interpolation it was ensured that both models had exactly the same amount of training data available to them and that the sizes of the kept and held-out sets were the same. In case of a back-off baseline model, it was trained on the whole training set, since such model does not require use of cross-validation for training. Both linear and log-linear interpolation models were used to smooth Katz back-off language model estimates.

For the smaller corpus, namely the Hub5 Switchboard corpus, multiple runs were performed and an attempt was made to completely characterise the relative performance of the log-linear interpolation with respect to the baseline models in terms of log-linear interpolation model parameters, namely the clustering criteria, and the training set sizes. It was discovered that for this corpus the log-linear interpolation performs almost as well as the linear interpolation, and in some cases slightly better, with the differences of perplexities being negligible. Both linear and log-linear interpolation models consistently outperformed the baseline Katz back-off model by 3.7%.

For the bigger Wall Street Journal corpus, the log-linear interpolation model

outperformed the linear interpolation by 2% to 2.6% and the back-off language model by 2.3% to 3.2% in terms of perplexity calculated over the three evaluation sets.

According to these findings it is possible to assume that log-linear smoothing of n-gram language models performs as well as linear interpolation on the small corpora and outperforms it on the bigger corpora, with both interpolation models achieving performance which is constistently superior to that of the unsmoothed back-off language model.

While an attempt has been made to systematically explore the log-linear smoothing for n-grams, there remain many directions that need to be explored, for example, it is interesting to see how the division of the training data into the kept and held-out sets (*i.e.* the ratio between the held-out and kept set sizes) affects the performance of log-linear language model. When the size of the held-out set is too small, the deterioration of the performance can be expected due to the fact that there is not enough data to estimate the interpolation weights, whereas if the kept set size is too small, the deterioration in the performance can be explained by the insufficient amount of data for obtaining the probability estimates to be smoothed.

CHAPTER 6

Summary and Conclusion

In this work a novel smoothing technique for n-gram language modelling, namely the log-linear interpolation was investigated. Log-linear interpolation of an n-gram language models is a technique which allows to form an exponential combination of probability estimates obtained for n-grams of different specificity trained on some text corpus. The resulting combination is guaranteed to perform no worse than the probability estimates comprising the model provided that these n-gram probability estimates are reliable enough, *i.e.* some form of discounting was used to obtain them in order to ensure that the probabilities are nonzero. Maximum likelihood estimates do not satisfy this condition and therefore are not considered in this work.

Log-linear interpolation was introduced by Klakow [39] who used it for combining different language models. This technique, interpolating the conventional n-grams and another type of language models called distance n-grams was successfully applied by Beyerlein *et al.* [5] to the broadcast news transcription task. Unlike the publications mentioned above, which essentially propose a general purpose method for combining the arbitrary language models, the aim of this work was to investigate the applicability of the technique to the special case of smoothing of n-gram probability estimates, and to attempt a rigorous derivation of theoretical framework within which the log-linear interpolation could be placed as a possible smoothing method. As a result, the framework for obtaining the log-linear interpolation probability estimates, including the techniques for parameter optimisation and clustering, was proposed.

In consequent experiments carried over the two popular language modelling corpora, the log-linear interpolation has shown to perform as well as linear interpolation when produced on a small corpus and outperform the linear interpolation on a big corpus, with both models outperforming the baseline back-off language model.

The log-linear interpolation of language models, as it has been shown in this work, seems to be a promising technique which can be successfully employed in the tasks of smoothing the language model probability estimates.

Bibliography

[1] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. A tree-based language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-37(7):1001–1008, July 1989.

[2] L. R. Bahl, P. F. Brown, P. V. de Souza, R. L. Mercer, and D. Nahamoo. A fast algorithm for deleted interpolation. In *Proceedings of the 2nd European Conference on Speech Communication and Technology*, volume 3, pages 1209–1212, Genova, September 1991.

[3] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. In A. Waibel and K. Lee, editors, *Readings in Speech Recognition*, chapter 6, pages 308–319. Morgan Kaufmann, San Mateo, CA, 1990. Reprinted from *'83 IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[4] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March 1996.

[5] P. Beyerlein, X. L. Aubert, R. Haeb-Umbach, M. Harris, D. Klakow, A. Wendemuth, S. Molau, M. Pitz, and A. Sixtus. The Philips/RWTH system for transcription of broadcast news. In *Proceedings of the 6th European Conference on Speech Communication and Technology*, volume 2, pages 647–650, Budapest, September 1999.

[6] E. Black, F. Jelinek, J. Lafferty, R. L. Mercer, and S. Roukos. Decision tree models applied to the labelling of texts with parts-of-speech. In *'92 DARPA Speech and Natural Language Workshop*, pages 117–121, 1992.

[7] G. E. P. Box and G. C. Tao. *Bayesian Inference in Statistical Analysis*. Addison-Wesley, Reading, MA, 1973. Section 1.3.

[8] T. Briscoe and J. Carrol. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59, 1993.

[9] P. F. Brown, V. J. Della Pietra, P. de Souza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[10] P. F. Brown, V. J. Della Pietra, R. L. Mercer, S. A. Della Pietra, and J. C. Lai. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40, March 1992.

[11] E. Charniak. *Statistical Language Learning*. Language, speech, and communication. MIT Press, Cambridge, MA; London, 1993.

[12] C. Chelba and F. Jelinek. Exploiting syntactic structure for language modeling. In *Proceedings of Association for Computational Linguistics*, volume ACL-36, Montreal, 1998.

[13] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of Association for Computational Linguistics*, volume ACL-34, pages 310–318, Santa Cruz, CA, June 1996.

[14] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, August 1998.

[15] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–394, October 1999.

[16] S. F. Chen, K. Seymore, and R. Rosenfeld. Topic adaptation for language modeling using unnormalized exponential models. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 681–684, Seattle, Washington, May 1998.

[17] K. W. Church and W. A. Gale. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5(1):19–54, January 1991.

[18] K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. In *Proceedings of Association for Computational Linguistics*, volume ACL-27, pages 76–83, 1989.

[19] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, 1991.

[20] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[21] W. N. Francis and H. Kučera. *Frequency Analysis of English Usage: Lexicon and Grammar*. Houghton-Mifflin, Boston, 1982.

[22] W. A. Gale and K. W. Church. Estimation procedures for language context: poor estimates are worse than none. In *COMPSTAT, Proceedings in Computational Statistics*, pages 69–74, Dubrovnik, Yugoslavia, September 1990.

[23] W. A. Gale and G. Sampson. Good-Turing frequency estimation without tears. *Journal of Quantitative Linguistics*, 2:217–237, 1995.

[24] J. Gao and X. Chen. Probabilistic word classification based on a context-sensitive binary tree method. *Computer Speech and Language*, 11(4):307–320, October 1997.

[25] M. Generet, H. Ney, and F. Wessel. Extensions of absolute discounting for language modeling. In *Proceedings of the 4th European Conference on Speech Communication and Technology*, volume 2, pages 1245–1248, Madrid, September 1995.

[26] J. J. Godfrey, E. C. Holliman, and J. McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 517–520, San Francisco, March 1992.

[27] I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3/4):237–264, December 1953.

[28] J. Goodman. Putting it all together: Language model combination. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 1647–1650, Istanbul, June 2000.

[29] M. Jardino. A class bigram model for very large corpus. In *Proceedings of International Conference on Spoken Language Processing*, volume 2, pages 867–870, Yokohama, September 1994.

[30] H. Jeffreys. *Theory of Probability*. Oxford Science Publications. Clarendon Press, Oxford, 3rd edition, 1961.

[31] F. Jelinek. Markov source modelling of text generation. In J. K. Skwirzynski, editor, *The Impact of Processing Techniques on Communication*. Martinus-Nijhoff Publishers, Dordrecht, 1985.

[32] F. Jelinek. Self-organised language modeling for speech recognition. In A. Waibel and K. Lee, editors, *Readings in Speech Recognition*, chapter 8, pages 450–503. Morgan Kaufmann, San Mateo, CA, 1990.

[33] F. Jelinek. *Statistical Methods for Speech Recognition*. Language, speech and communication. MIT Press, Cambridge, MA; London, March 1997.

[34] F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*. North-Holland, Amsterdam, May 1980.

[35] F. Jelinek and R. L. Mercer. Probability distribution estimation from sparse data. *IBM Technical Disclosure Bulletin*, (28):2591–2594, 1985.

[36] I. Kanter and D. A. Kessler. Markov processes: Linguistics and Zipf's law. *Physical Review Letters*, 74(22):4559–4562, May 1995.

[37] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):400–401, March 1987.

[38] C. T. Kelley. Detection and remediation of stagnation in the Nelder–Mead Algorithm using a sufficient decrease condition. *Society for Industrial and Applied Mathematics Journal on Optimization*, 10(1):43–55, 1999.

[39] D. Klakow. Log-linear interpolation of language models. In *Proceedings of 5th International Conference on Spoken Language Processing*, volume 5, pages 1695–1699, Sydney, December 1998.

[40] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 181–184, Detroit, May 1995.

[41] H. Kučera and W. N. Francis. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI, 1967.

[42] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder–Mead Simplex method in low dimensions. *Society for Industrial and Applied Mathematics Journal on Optimization*, 9(1): 112–147, 1998.

[43] P. S. Laplace. *Philosophical Essay on Probabilities*, volume 13 of *Sources in the history of mathematics and physical sciences*. Springer-Verlag, New York, 1995. Translated from the 5th French edition of 1825.

[44] E. L. Lehmann. *Theory of Point Estimation*. Probability and mathematical statistics. John Wiley & Sons, New York, 1983.

[45] W. Li. Random texts exhibit Zipf's-law-like word frequency distribution. *IEEE Transactions on Information Theory*, 38(6):1842–1845, 1992.

[46] W. Li. Comments on Zipf's law and the structures and evolution of natural languages. *Complexity*, 3(5):9–10, 1998. Letters to the editor.

[47] D. Magerman. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. dissertation, Stanford University, February 1994. Section 3.4.2.

[48] B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman, New York, 1983.

[49] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. Language, speech, and communication. MIT Press, Cambridge, MA; London, August 1999. Section 1.4.3.

[50] S. C. Martin, C. Hamacher, J. Liermann, F. Wessel, and H. Ney. Assessment of smoothing methods and complex stochastic language modeling. In *Proceedings of the 6th European Conference on Speech Communication and Technology*, volume 5, pages 1939–1942, Budapest, September 1999.

[51] S. C. Martin, H. Ney, and J. Zaplo. Smoothing methods in maximum entropy language modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume I, pages 545–548, Phoenix, March 1999.

[52] D. D. McDonald. Internal and external evidence in the identification and semantic categorization of proper names. In B. Boguraev and J. Pustejovsky, editors, *Corpus Processing for Lexical Acquisition*, Language, speech, and communication, pages 21–39. MIT Press, Cambridge, MA; London, 1995.

[53] K. I. M. McKinnon. Convergence of the Nelder–Mead Simplex method to a nonstationary point. *Society for Industrial and Applied Mathematics Journal on Optimization*, 9(1):148–158, 1998.

[54] A. Nadas. On Turing's formula for word probabilities. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-33(6):1414–1416, December 1985.

[55] H. Ney and U. Essen. On smoothing techniques for bigram-based natural language modelling. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 825–828, Toronto, May 1991.

[56] H. Ney and U. Essen. Estimating 'small' probabilities by leaving-one-out. In *Proceedings of the 3rd European Conference on Speech Communication and Technology*, volume 3, pages 2239–2242, Berlin, September 1993.

[57] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependencies in stochastic language modeling. *Computer Speech and Language*, 8(1): 1–28, January 1994.

[58] H. Ney, S. Martin, and F. Wessel. Statistical language modeling using leaving-one-out. In S. Young and G. Bloothooft, editors, *Corpus-Based Methods in Language and Speech Processing*, chapter 6, pages 174–207. Kluwer Academic Publishers, Dordrecht, 1997.

[59] T. Niesler. *Category-based statistical language models*. Ph.D. dissertation, Cambridge University, June 1997.

[60] D. B. Paul and J. M. Baker. The design for the Wall Street Journal-based CSR corpus. In *Proceedings of International Conference on Spoken Language Processing*, pages 899–902, 1992.

[61] L. C. Bauman Peto. *A Comparison of Two Smoothing Methods for Word Bigram Models*. M.Sc. dissertation, Graduate Department of Computer Science, University of Toronto, October 1994.

[62] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1986.

[63] E. S. Ristad. A natural law of succession. Research Report CS-TR-495-95, Department of Computer Science, Princeton University, July 1995.

[64] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. dissertation, Carnegie Mellon University, April 1994. Published as Research Report CMU-CS-94-138. Appendix B.

[65] C. Samuelsson. Relating Turing's formula and Zipf's law. Research Report CLAUS-78, Computational Linguistics Department, Universität des Saarlandes, Saarbrücken, June 1996.

[66] C. E. Shannon. Prediction and entropy of printed English. *The Bell System Technical Journal*, 30:50–64, January 1951.

[67] H. S. Sichel. On a distribution law for word frequencies. *Journal of the American Statistical Association*, 70(351):542–547, September 1975.

[68] Z. K. Silagadze. Citations and the Zipf-Mandelbrot's law. *Complex Systems*, 11(6), 2000.

[69] R. F. Simmons and Y. Yu. The acquisition and use of context-dependent grammars for English. *Computational Linguistics*, 18(4):391–418, December 1992.

[70] A. A. Tsonis, C. Schultz, and P. A. Tsonis. Zipf's law and the structure and evolution of languages. *Complexity*, 2(5):12–13, 1997.

[71] F. Wessel and A. Baader. Robust dialogue-state dependent language modeling using leaving-one-out. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume II, pages 741–744, Phoenix, March 1999.

[72] P. C. Woodland, M. J. F. Gales, D. Pye, and V. Valtchev. The HTK large vocabulary recognition system for the 1995 ARPA H3 task. In *Proceedings of the ARPA Speech Recognition Workshop*, New York, 1996. Harriman House.

[73] P. C. Woodland, C. J. Leggetter, J. J. Odell, V. Valtchev, and S. J. Young. The 1994 HTK large vocabulary speech recognition system. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 73–76, Detroit, May 1995.

[74] G. K. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, Reading, MA, 1949.

[75] G. K. Zipf. *The Psycho-Biology of Language: An Introduction to Dynamic Philology*. MIT Press, Cambridge, MA, 1968. Reprint of Houghton-Mifflin edition of 1935.