On the Estimation of Discount Parameters for Language Model Smoothing

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney

Human Language Technoloy and Pattern Recognition,
Computer Science Department, RWTH Aachen University
{sundermeyer, schlueter, ney}@cs.rwth-aachen.de

Abstract

The goal of statistical language modeling is to find probability estimates for arbitrary word sequences. To obtain non-zero values, the probability distributions found in the training data need to be smoothed. In the widely-used Kneser-Ney family of smoothing algorithms, this is achieved by absolute discounting. The discount parameters can be computed directly using some approximation formulas minimizing the leaving-one-out log-likelihood of the training data.

In this work, we outline several shortcomings of the standard estimators for the discount parameters. We propose an efficient method for computing the discount values on held-out data and analyze the resulting parameter estimates. Experiments on large English and French corpora show consistent improvements in perplexity and word error rate over the baseline method. At the same time, this approach can be used for language model pruning, leading to slightly better results than standard pruning algorithms.

Index Terms: language model smoothing, absolute discounting, Kneser-Ney method, language model pruning

1. Introduction

The task of statistical language modeling consists of estimating accurate probabilities for any possible word sequence. Many different modeling approaches exist, but only few of them show competitive results in real-world applications. Recent advances include improvements in exponential models ('model M', [1]) and neural networks ([2], [3]) where feed-forward or recurrent network architectures are trained to estimate word posterior probabilities.

On the other hand, standard n-gram smoothing techniques for language modeling are still superior in several aspects: Besides their good performance in terms of word error rate, they are essential for efficient decoding, and due to fast training times, they can benefit from arbitrarily large amounts of training data. In addition, any modeling approach can usually be improved by interpolation with a smoothed n-gram language model (LM), see e. g. [3].

A variety of smoothing algorithms has been proposed, among which the Kneser-Ney method ([4]) often has been reported to give best results.

This technique relies on absolute discounting. Given an n-gram (h, w) where w denotes a word, and h the preceding history words, a constant offset b (discount) is subtracted from the n-gram count $N(h, w)$ found in the training data. Interpolating higher-order probability estimates with lower-order ones, this results in the estimation formula

$$p(w|h) = \max\left\{\frac{N(h,w) - b}{N(h,\cdot)}, 0\right\} + \gamma(h) \cdot p(w|\bar{h}).$$

We stick to the notation introduced in [4]: By $\bar{h}$ we denote the generalized history, dropping the leftmost history word, and $\gamma(h)$ is the interpolation weight. The above recursion terminates at the unigram level, interpolating unigram estimates with zerogram probabilities.

In [4], the method of absolute discounting is refined by introducing an optimized lower-order distribution, using so-called modified counts. Based on this work, [5] and [6] develop smoothing methods for multiple discounts (modified Kneser-Ney smoothing). Then, in the above formula, b needs to be replaced by $b_{N(h,w)}$ as the discount depends on the n-gram count.

For all smoothing algorithms discussed here, approximation formulas have been derived to compute the values of the discount parameters. In [6], the authors also compute the discount parameters based on held-out data. Unfortunately, they optimize parameters just one by one using Powell's algorithm. They give results for at most three parameters. Their experiments seem to suggest that no improvements can be obtained by using a held-out corpus when the amount of training data is large.

In addition, it is common practice not to optimize the discounts: Standard software implementations like [7] and [8] only offer the direct computation of the discount values based on conventional formulas.

In this work, we address the following two questions:

1. What is the impact of using discount values which are optimized with respect to perplexity when a large amount of training data is available?
2. How can these values be computed efficiently?

Contrasting previous experiments, we also extend the number of discounts to much larger values. We run experiments on several corpora, reporting on results in terms of perplexity and word error rate. Finally, we show that our method can easily be applied to language model pruning.

2. Estimation of Discount Parameters

The standard discount equations are derived in such a way that the leaving-one-out log-likelihood of the training data

$$\sum_{hw} N(h,w) \cdot \log p_1(w|h)$$

is minimized. Here, $p_1(w|h)$ denotes the probability estimate of w given h, leaving out one occurrence of the n-gram (h, w) from the training data.

In case of a single parameter b, this results in the approximate discount estimator $b = n_1/(n_1 + 2n_2)$, where n_r denotes the number of n-grams whose count (or modified count) is equal to r in the training data. Often, three discounts b_1, b_2, and b_{3+} are used, depending on whether $N(h, w) = 1$, $N(h, w) = 2$, or $N(h, w) \geq 3$. Then, the discounts are usually set according to the equations

$$b_1 = 1 - 2b\frac{n_2}{n_1} = b, \quad b_2 = 2 - 3b\frac{n_3}{n_2}, \quad b_{3+} = 3 - 4b\frac{n_4}{n_3}.$$

Using these formulas for computing the discounts may incur the following problems:

1. The above equations are (and can only be—no closed-form solutions exist) only approximations to the exact solutions.
2. Their derivations depend on backing-off smoothing (see below for more details). On the other hand, interpolation generally performs better. Nevertheless, discounts of interpolated models are computed using the backing-off discount formulas.
3. It is not clear how to set the discounts when a language model is pruned. In practice, the discounts are computed for the full model, and are not changed after pruning.

Furthermore, the derivation of the lower-order distribution in [4] and [6] relies on using a single discount parameter. Using more than one parameter results in probability distributions that do not satisfy the marginal constraints exactly (see e. g. [9]). On the other hand, increasing the number of discounts results in a more accurate modeling of the probability mass for unseen events, and experiments show consistent improvements.

For these reasons, we revisit the idea of optimizing the discounts on held-out data. In principle, the optimization could also be carried out on the training data using leaving-one-out, but then the optimization process would be overly complex as the amount of training data usually exceeds the size of held-out data by several orders of magnitude.

2.1. Interpolation

The Kneser-Ney smoothing algorithm makes use of the same formula as absolute discounting for the estimation of $p(w|h)$. The interpolation weight $\gamma(h)$ is defined as

$$\gamma(h) = \frac{b_{|b|+}N_{|b|+}(h, \cdot) + \sum_{r=1}^{|b|-1} b_r N_r(h, \cdot)}{N(h, \cdot)}$$

where $|b|$ is the number of discounts, and $N_r(h, \cdot)$ is the number of words that have been observed r times with the given history.

The log-likelihood function F on held-out data can then be defined as follows:

$$F = \sum_{hw} C(h, w) \log p(w|h)$$

where the $C(h, w)$ denote the count of (h, w) in the held-out data. Normally, discount parameters are not tied across different orders, resulting in a matrix $B \in \mathbb{R}^{n\times|b|}$ of discount parameters for n-gram models.

Now we can compute the optimal discount values by maximizing F. The corresponding optimization problem is not concave ([10]): Even when restricting to bigrams (v, w) and a single discount, after some calculations, for the eigenvalues $\lambda_{1,2}$ of the Hessian of $p(w|v)$ we obtain

$$\lambda_{1,2} = \pm N_{1+}(v, \cdot)\frac{W - N_{1+}^{+}(\cdot, \cdot)}{WN^{+}(\cdot, \cdot)N(v, \cdot)}$$

where the superscript $+$ indicates modified counts, and W is the vocabulary size. Because of opposite signs for $\lambda_{1,2}$, the Hessian is not negative-semidefinite, and thus $p(w|h)$ not concave. Finally, as $p(w|h)$ is not concave, $\log p(w|h)$ is neither.

We optimize F using the improved Resilient Propagation algorithm (RPROP) including weight backtracking as described in [11] which was shown to yield good performance when optimizing non-convex (or, in our case, non-concave) problems, and which is easy to implement.

As RPROP is a first-order optimization method, we need to compute ∇F. For the discount r of a given order, we have

$$\frac{\partial F}{\partial b_r} = \sum_{hw} C(h, w)\frac{1}{p(w|h)}\frac{\partial}{\partial b_r}p(w|h),$$

and subsequently, for the highest order discounts we obtain

$$\frac{\partial}{\partial b_r}p(w|h) = \frac{N_r(h, \cdot)}{N(h, \cdot)}p(w|\bar{h}) - \begin{cases} 1/N(h, \cdot), & N(h, w) = r \\ 0, & \text{else} \end{cases}$$

given that $N(h, \cdot) > 0$. In case $N(h, \cdot) = 0$, the derivative is zero. For brevity, we omit the derivatives for the lower-order discounts. The general structure of the above derivative remains the same, except for multiplying higher-order interpolation weights.

2.2. Backing-off

In principle, the same approach can be done for backing-off smoothing. In this case, the probability estimate for $p(w|h)$ takes the form

$$p(w|h) = \begin{cases} \dfrac{N(h, w) - b_{N(h,w)}}{N(h, \cdot)} & N(h, w) > 0 \\ \tilde{\gamma}(h) \cdot p(w|\bar{h}) & \text{else.} \end{cases}$$

The problem is that the backing-off weight

$$\tilde{\gamma}(h) = \gamma(h) \Big/ \left(1 - \sum_{w':N(h,w')>0} p(w'|\bar{h})\right)$$

now also includes a normalization term which cannot be dropped in the derivative for lower-order discounts. We thereby lose an important advantage of the optimization for interpolation: The complexity of the target function F only depends on the size of the held-out corpus. In contrast, due to the summation over all words from the training corpus, for backing-off this no longer is the case.

As backing-off models usually perform worse than interpolated ones, we decided not to investigate the optimization of backing-off models in more detail.

2.3. Pruning

A simple way of reducing the size of a language model is to use count-cutoffs: All n-grams seen less than a predefined threshold are discarded. As a result, the probability $p(w|h)$ for pruned n-grams (h, w) is modeled by the lower-order distribution.

An alternative approach to LM pruning based on relative entropy was introduced in [15]. In contrast to count-cutoffs, this method is self-contained, i. e., the pruning method does not rely on the count data but only on the LM itself. Let p be the probability distribution of an LM, and let p' be the distribution p where the estimate $p(w|h)$ has been removed. The entropy criterion then discards those n-grams for which

$$p(h)\sum_{w} p(w|h)\frac{\log p(w|h)}{\log p'(w|h)} < \theta$$

for a predefined threshold θ.

				NAB		Quaero EN		Quaero FR	
Smoothing	$\|b\|$	type	count-cutoffs	dev	test	dev	test	dev	test
KN	1	backing-off	1-1-1-1	90.2	91.4	226.8	226.4	177.4	199.3
KN	1	interpolated	1-1-1-1	86.1	87.2	220.0	218.6	167.7	191.2
mod KN	3	interpolated	1-1-1-1	**84.5**	**85.7**	210.9	209.2	161.9	183.2
mod KN	3	interpolated	1-1-1-2	89.0	90.3	**208.5**	**206.1**	**160.5**	**181.6**
mod KN	3	interpolated	1-1-1-1-1	78.3	79.3	208.4	207.0	160.0	180.8
opt KN	1	interpolated	1-1-1-1	85.7	86.8	212.4	210.5	163.0	185.1
opt KN	3	interpolated	1-1-1-1	84.4	85.5	203.1	200.7	157.8	177.4
opt KN	3	interpolated	1-1-1-1-1	77.9	78.9	200.0	197.9	155.9	175.2
opt KN	10	interpolated	1-1-1-1	84.3	85.4	200.9	198.5	156.8	175.5
opt KN	10	interpolated	1-1-1-1-1	77.9	78.9	200.3	197.6	155.0	173.5
opt KN	50	interpolated	1-1-1-1	**84.3**	**85.4**	**199.5**	**197.9**	**156.1**	**175.2**

Table 1: Perplexity results for different smoothing methods, number of discount parameters, and count-cutoffs; a count-cutoff of '1-1-1-2' means that at the fourgram level, only n-grams seen at least twice are retained, whereas singletons are kept for lower orders.

In [9] and [16] it was observed that this algorithm does not interact well with Kneser-Ney (KN) smoothing. For this reason, in [17] a refinement was proposed: The quantity $p(h)$ is estimated by a Katz-smoothed LM.

For both approaches, the discount parameters are estimated on the full model, and they are kept constant after pruning. On the other hand, neither the formula-based estimates nor the optimized discount values are appropriate for the pruned model because it is assumed that all n-grams are used for the LM estimation.

For this reason, we propose to include the pruning regime into the discount optimization process. In our optimization framework, this can be easily achieved in combination with count-cutoffs: For example, when discarding singletons, this corresponds to setting $b_1 = 1$. As a result, in interpolated KN smoothing only the lower-order estimate $\gamma(h) \cdot p(w|\bar{h})$ is non-zero. All remaining discounts are optimized.

3. Experimental Results

To evaluate our method of optimizing the discounts, we ran a series of experiments on three different corpora for English (North American Business Task (NAB), and Quaero[1]), and French (Quaero). These are described in Table 2.

Corpus	set	words	OOV	domain
NAB EN	train	247.4 M	0.86 %	newspaper
	dev	919.1 K	0.86 %	
	test	914.7 K	0.86 %	
Quaero EN	train	348.0 M	1.28 %	blog+forum transcriptions
	dev	41.8 K	0.45 %	
	test	1.2 M	0.49 %	
Quaero FR	train	243.4 M	1.15 %	blog+forum transcriptions
	dev	46.7 K	0.45 %	
	test	700.4 K	0.01 %	

Table 2: Statistics for the training, development and test data for the experiments (OOV = out-of-vocabulary rate); the vocabulary sizes are 64 K, 150 K, and 200 K for NAB, Quaero English and French, respectively.

For our experiments, we intentionally did not use LM interpolation because this might mask discount optimization effects. For the LM of the full recognition systems described in [12], we also used data from other sources. Among these sources, the Quaero blog data were found to match the test data best in terms of LM interpolation weight.

[1] Quaero research programme, see http://www.quaero.org.

The results regarding perplexity are summarized in Table 1. We also give perplexity values for the development data to see if the improvements obtained when optimizing on development data translate to the test data. We observe that this is the case, even when optimizing a large number of discounts, relative improvements closely match on both sets. This means that optimizing the discounts does not produce overfitting on the development data.

In general, optimizing discounts on held-out data as well as using more discounts helps for all LM corpora investigated here. While the improvements in perplexity are only small for NAB, they are comparatively large for the Quaero corpora. For English, the perplexity could be decreased from 209.2 to 197.9, for French from 161.9 to 156.1. Careful tuning of count-cutoffs may slightly reduce this difference: While LMs trained using the standard formulas may benefit from discarding singletons (see rows on modified (mod) KN), the converse is true for optimized discounts (opt KN, not shown in the table). The fact that more data—in the form of singletons—do not pay off in terms of perplexity indicates that using the approximation formulas does not produce appropriate values.

For all corpora, optimized discount values significantly differ from those predicted by the formulas. For highest order discounts, on NAB we obtained an optimized value of $b_{3+} = 1.70$, as opposed to 1.50 for the standard discount formula. For the English Quaero data, these differences are even larger (1.32 vs. 2.55), and they seem to depend on the complexity of the training and test data as indicated by absolute perplexity values.

Our optimized discounts are in accordance with previous assumptions stated in [13] concerning the *monotonicity constraints* $r - b_r \leq r + 1 - b_{r+1}$ which we found usually to be fulfilled if $|b|$ not too large. On the other hand, our values contradict the *interval constraints* from [14], requiring $0 < b_r < 1$.

Due to the non-convexity of the optimization problem, in principle it is possible that a different set of discount parameters exists, yielding even better perplexity values.

For Quaero English, we also created lattices for the development and evaluation data 2010 (41.0 K running words) using the state-of-the-art acoustic models of the single best system described in [12] and the 10-discount 1-1-2-2 LM. We then applied an LM rescoring step using the modified KN 1-1-1-2 and the opt KN 50 1-1-1-1 LMs. We obtained slight improvements in word error rate of $0.2\,\%$ absolute on both the development and evaluation corpora (see Table 3). In practice, improvements may be higher than those reported here since the two different LMs were only applied in the LM rescoring step, but not in the first recognition passes.

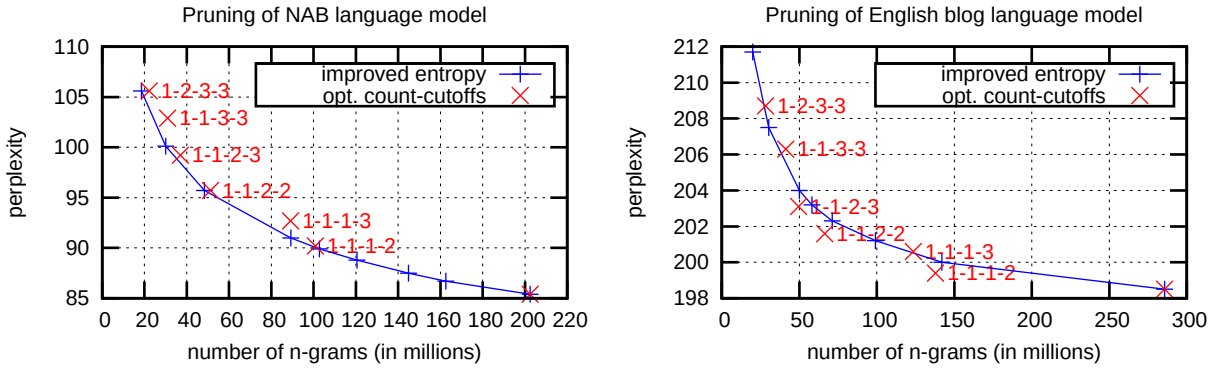

Figure 1: Pruning results for count-cutoffs including optimized discounts vs. improved entropy criterion

Smoothing	cutoffs	dev	eval
mod KN 3	1-1-1-2	21.2 %	20.7 %
opt KN 50	1-1-1-1	21.0 %	20.5 %

Table 3: Word error rate results for Quaero EN (LM rescoring).

Figure 1 shows results for LM pruning based on the 4-gram LMs using 10 discount parameters. For the NAB corpus (left hand side), it can be seen that the count-cutoff performance is very close to the one of the improved entropy criterion, especially when pruning singletons only.

For Quaero English (right hand side), most of the time optimized count-cutoffs perform even better than the entropy criterion. This may be due to the fact that the trigram Katz LM used for the improved entropy pruning cannot be adapted on held-out data. On the other hand, experiments seem to indicate that it is more important to optimize the discounts after pruning than to choose n-grams for pruning based on relative entropy. At the same time, the count-cutoff approach is much simpler and does not require the estimation of a second LM. (We do not show results for French as these are similar in nature to those for Quaero English.)

Finally, it should be noted that in case of the Quaero corpora, the amount of development data is quite small. Furthermore, the optimization process can be done very efficiently. In our experiments, the target function always converged within less than 100 iterations. For the Quaero development set and 10 discounts, this takes about 80 seconds on a standard CPU core.

4. Conclusions

In this paper, we investigated the optimization of discount parameters for large amounts of training data. Experiments show that increasing the number of parameters and optimizing them on held-out data can improve perplexity by up to 5 % relative, also affecting word error rates. Second, the optimization of discounts is relevant to LM pruning: Often, the entropy pruning method is slightly outperformed by count-cutoffs as discounts are not optimized after pruning.

The discount optimization already works well for small amounts of in-domain development data which are usually available in speech recognition. In our experiments, the perplexity on test data could always be improved, and it should be stressed that this comes at negligible computational costs. Therefore, we conclude that optimized discounts are favorable to the standard estimation formulas.

Acknowledgement: This work was partly realized as part of the Quaero programme, funded by OSEO, French State agency for innovation.

5. References

[1] Chen, S. F., "Shrinking Exponential Language Models", Proc. of HLT-NAACL 2009, pp. 468–476

[2] Schwenk, H., "Continuous space language models", Computer Speech and Language 21 (2007), pp. 492–518

[3] Mikolov, T., Karafiát, M., Burget, L., Černoczký, J. H., and Khudanpur, S., "Recurrent neural network based language model" Proc. of Interspeech 2010, pp. 1045–1048

[4] Kneser, R., and Ney, H., "Improved Backing-Off For M-Gram Language Modeling", Proc. of ICASSP 1995, pp. 181–184

[5] Ney, H., Martin, S., and Wessel, F., "Statistical Language Modeling Using Leaving-One-Out", in Young, S., and Bloothooft, G., "Corpus-Based Methods in Language And Speech Processing", Kluwer Acad. Publ. 1997

[6] Chen, S. F., and Goodman, J., "An Empirical Study of Smoothing Techniques for Language Modeling", Harvard University, Tech. Rep. TR-10-98, August 1998

[7] Stolcke, A., "SRILM—An Extensible Language Modeling Toolkit", Proc. of ICSLP 2002, pp. 901–904

[8] Federico, M., Bertoldi, N., and Cettolo, M., "IRSTLM: An Open Source Toolkit for Handling Large Scale Language Models", Proc. of Interspeech 2008, pp. 1618–1621

[9] Siivola, V., Hirsimäki, T., and Virpioja, S., "On Growing and Pruning Kneser-Ney Smoothed N-Gram Models", IEEE Trans. on Audio, Speech, and Language Processing, Vol. 15, No. 5, July 2007

[10] Boyd, S., and Vandenberghe, L., "Convex Optimization", Cambridge University Press 2004

[11] Igel, C., and Hüsken, M., "Empirical evaluation of the improved Rprop learning algorithm", Neurocomputing 50 (2001), pp. 105–123

[12] Sundermeyer, M., Nußbaum-Thom, M., Wiesler, S., Plahl, C., El-Desoky Mousa, A., Hahn, S., Nolden, D., Schlüter, R., and Ney., H., "The RWTH 2010 Quaero ASR Evaluation System for English, French, and German", Proc. of ICASSP 2011, pp. 2212–2215

[13] Andrés-Ferrer, J., Ney., H., "From empirical Bayes to Leaving-one-out", Proc. of IWSM 2011, accepted for publication

[14] Andrés-Ferrer, J., Ney., H., "Extensions of Absolute Discounting (Kneser-Ney Method)" Proc. of ICASSP 2009, pp. 4729–4732

[15] Stolcke, A., "Entropy-based Pruning of Backoff Language Models", Proc. of ICSLP 2002, pp. 901–903

[16] Chelba, C., Brants, T., Neveitt, W., Xu, P., "Study on Interaction between Entropy Pruning and Kneser-Ney Smoothing" Proc. of Interspeech 2010, pp. 2422–2425

[17] Stolcke, A., SRILM user mailing list, September 28, 2010