Growing an n-gram language model

Vesa Siivola[1] *and Bryan L. Pellom*[2]

[1]Neural Networks Research Centre, Helsinki University of Technology, Finland
[2]The Center for Spoken Language Research, University of Colorado at Boulder, USA
vesa.siivola@hut.fi, pellom@cslr.colorado.edu

Abstract

Traditionally, when building an n-gram model, we decide the span of the model history, collect the relevant statistics and estimate the model. The model can be pruned down to a smaller size by manipulating the statistics or the estimated model. This paper shows how an n-gram model can be built by adding suitable sets of n-grams to a unigram model until desired complexity is reached. Very high order n-grams can be used in the model, since the need for handling the full unpruned model is eliminated by the proposed technique. We compare our growing method to entropy based pruning. In Finnish speech recognition tests, the models trained by the growing method outperform the entropy pruned models of similar size.

1. Introduction

N-grams are the most widely used method for language modeling in speech recognition systems. The size of the language model grows rapidly with growing n, so it is common to choose a relatively low model order ($n = 3 \dots 5$). There are methods to prune the estimated model to a smaller size with reasonable reduction in the modeling accuracy. For example with cut-offs, we throw away all n-gram counts not exceeding a certain threshold and estimate the model based on the remaining counts.

A more principled approach to model pruning is entropy based pruning [1]. For each n-gram in the model, this method calculates how much the training set likelihood would decrease, if the n-gram was removed from the model. If this reduction does not exceed a given threshold, the n-gram is removed. This method is used as the baseline method in our experiments.

The pruning methods start with a full model containing also the n-grams that will be eventually pruned off. We present a method that starts from a unigram model and only adds the n-grams that will be needed in the final model. The advantage is that we do not need to set the highest possible order of n-grams and can include also reasonable amount of very high order n-grams if necessary. Also the model construction takes less memory. On the other hand, the model estimation has to be carefully implemented to make it reasonably fast.

The first tests of the growing method were conducted on Finnish corpora [2]. In this paper, we report recognition experiments on different Finnish audio data and with new acoustic models and decoder. We also present initial results on the U.S. English Broadcast News task. The English cross-entropy tests were run on different training set sizes in order to evaluate how the methods cope with differing amounts of training data.

2. Growing the n-gram model

The growing method is inspired by the Minimum Description Length principle [3], where the object is to compress the given data so that the number of bits required to send a model of the data and the actual data given the model is minimized. In this case, we want to minimize the sum of log likelihood of the training data and the model size.

We start with an initial model $\mathcal{M}_{old}$. We build the model by drawing an (m-1)-gram g_{m-1} from some distribution (discussed later). We find all the m-grams g_m from the training data that start with the prefix g_{m-1} and add this set G to the model, if the change Δ in data coding length is negative:

$$\Delta = (\alpha S_{new} - \log L_{new}) - (\alpha S_{old} - \log L_{old}) \,, \quad (1)$$

where S is the model size, that is *the model coding length*, and L is the training data likelihood. This procedure is iterated until some stopping criterion is met. The coefficient α was added to scale the relative importance of the training set likelihood compared to the model size. Since we are not actually wanting to code the training set data, we can change α to create models of different sizes.

Since we add the n-grams to the model in sets, it is useful perform final fine tuning of the model by trimming away the individual n-grams that do not decrease the data coding length. This last step is practically identical to the entropy based pruning and helps to create significantly better small models.

2.1. The likelihood of the training data

The likelihood L of the data is simply the probability of the training data:

$$L = \prod_i P(w_i | w_{i-1}, \dots) \quad (2)$$

Let us define the set I, that contains the indices of the words, for which the training set probabilities of the old (P_{old}) and new (P_{new}) models differ:

$$I = \{ j \mid P_{new}(w_j | w_{j-1}, \dots) \neq P_{old}(w_j | w_{j-1}, \dots) \} \quad (3)$$

The change in the data log likelihood is then

$$\log L_{new} - \log L_{old} = \sum_{i \in I} \log \frac{P_{new}(w_i | w_{i-1}, \dots)}{P_{old}(w_i | w_{i-1}, \dots)} \quad (4)$$

As we only need to calculate the change in the log likelihood, we can leave out most of the computations.

2.2. The model coding length

This section describes a hypothetical efficient coding of the language model. This gives us an estimate of the model size for the algorithm. In practice, during the model estimation, it is useful to store the model in a structure that is easier to work with.

We are actually only interested in the change of the model size. Thus, assuming our vocabulary V is constant, we are not interested in how it can be coded. For simplicity, we assume that the number of the n-grams in the model is coded as a 32-bit integer. It takes equal space in all models and can also be ignored. In addition to the vocabulary, we need to store the n-grams. We enumerate the words of the vocabulary from zero upwards. Thus, we can encode any word as an integer in $\log_2 |V|$ bits.

We need to encode which words actually make up the n-gram. There is an efficient tree structure for storing the n-grams, which exploits the natural sparsity and structure of n-gram models [4]. The tree contains all the n-grams of the model, regardless of the order of the n-gram. Each node stores only the index of the last word of the n-gram. To find an n-gram, one would start from the (empty) root, look for the child node that has the index of the first word of the gram, proceed to that node and look for the child node containing the index of the next word of the n-gram and so on until all of the indices are consumed. The node where the process stops is the desired node. Thus, for storing indices of the n-gram, it is sufficient to store only one index to the tree. For simplicity, we assume that there is always a path in the tree to the desired node and all nodes can have child nodes. The tree can be efficiently stored in a vector [4]: each node stores the range, where its child nodes can be found. Actually, it is only necessary to store the start of the range, since the end can be deduced from the start of the next node's range. The cost of this bookkeeping is thus $\log_2 N$ bits, where N is the number of n-grams in the model. All in all, the cost of storing the n-grams to the tree sturcture is $N \log_2(N \cdot |V|)$ bits.

What is left is to store the actual parameters of an n-gram. For each n-gram g_n, we need to store the corresponding probability. The interpolation (or back-off) coefficient is the same for all n-grams g_n which start with the prefix g_{n-1} and needs to be stored. We make the assumption that all n-grams are prefixes to some (n+1)-grams and each n-gram thus has space allocated for storing both the probability and the interpolation (or back-off) coefficient. The cost of storing these two floating point numbers is 2θ, where θ is the number of bits needed for the desired precision.

Finally, we see that the model coding length is

$$S = N\left(\log_2(N \cdot |V|) + 2\theta\right) = N\left(C + \log_2(N)\right) \qquad (5)$$

where C is a constant. The difference in model coding lengths is:

$$\Delta S = (N_{new} - N_{old})\, C + N_{new} log_2 N_{new} - N_{old} log_2 N_{old} \quad (6)$$

If we would ignore the terms $N_{new} log_2 N_{new}$ and $N_{old} log_2 N_{old}$, we would get something closely resembling the Bayesian Information Criterion. In our experiments, ignoring these terms did not significantly change the cross-entropy results, but hurt the speech recognition results.

2.3. Practical issues: putting it together

We have given the mathematical theory behind the algorithm for adding n-grams to the model. In this section, we will fill in the practical details. The model building was started from a unigram model. Other choices are possible. When looking for new n-grams of order m, the algorithm first samples a (m-1)-gram prefix from some distribution. We have simply chosen to sample the model which we are building, n-gram by n-gram from unigrams up to the highest order of n-gram that eventually gets added to model. Incidentally, this kind of sampling assures, that in the n-gram tree there is always a prefix path up to the current n-gram, making an earlier assumption true. The stopping criterion for the algorithm comes naturally: when all the n-grams of the model have been used as prefixes, we are done.

Our sampling processes one n-gram order at a time. Since we know which prefixes the sampling will use, we do not need to compute the statistics for the full set of n-grams. These statistics can be collected before processing the n-gram order. This algorithm is significantly faster than our older method, in which multiple binary searches were performed [2].

To add a chosen set G of n-grams to the model, we modify the necessary statistics. If we were using absolute discounting, this would be simple: we would only need to increment the counts of the corresponding n-grams. We have decided to use modified interpolated Kneser-Ney smoothing, which has been shown to generally outperform the other well known smoothing methods [5]. Therefore, we also have to modify the lower order n-gram counts accordingly. The discount coefficients of the smoothing are optimized numerically using Powell search [6]. The discount coefficients are re-estimated each time 10 000 new sets of n-grams of a new order have been accepted.

3. Experiments

3.1. Data

The Finnish text data was a collection of books and newspaper articles from the Finnish language bank [7] and short news stories from the Finnish news agency totaling 36M words. Earlier, we have experimented with different ways of splitting words to smaller units (no split, statistical morphs, syllables, grammatical morphemes) and found that splitting words to statistical morphs generally outperforms the other methods [8, 9]. This splitting results in 100M morphs (26k different morphs). The Finnish audio data was a book read by one female speaker (13h).

The North American News Text Corpus as well as the transcripts from 1996 Hub4 evaluation (LDC97T22) were used for training the English language models (520M words). 64k most common words were chosen as the vocabulary of the recognizer. For training the English system, 104 hours of the 1996 Broadcast News Speech Corpus (LDC97S44) and 97 hours of the 1997 Broadcast News Speech Corpus (LDC98S71) were used. For testing we used the DARPA 1998 BN Evaluation data (LDC2000S86). Specifically, we evaluated the system performance on the labeled F0 condition of the 1998 BN eval set with hand-labeled segmentation and speaker labels as reference (i.e., without an automatic segmentation and labeling software).

3.2. Speech recognition systems

For Finnish recognition experiments, we used the speech recognition system of the Neural Networks Research Centre (NNRC) [10]. For English we used the SONIC system [11, 12], which is freely available for research purposes from the Center for Spoken Language Research website. Both systems share similar architecture and are based on decision-tree state-clustered hidden Markov triphone models with continuous density mixture-Gaussians. Each clustered state is additionally associated with a gamma probability density function to model state durations. Both systems have a decoder that implements an efficient time-synchronous, beam-pruned Viterbi token-passing search through a static reentrant lexical prefix tree. We point out that the NNRC decoder has been designed to efficiently handle higher order n-grams in the first pass to provide improved recognition performance in languages such as Finnish. SONIC,

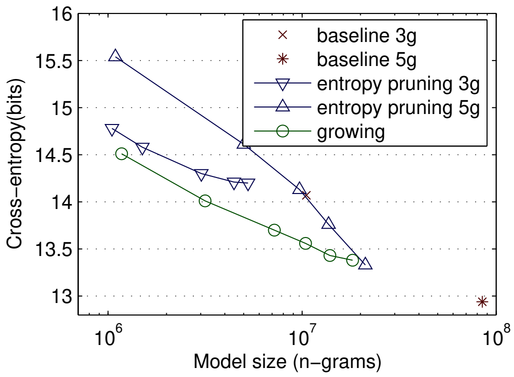

Figure 1: *Finnish cross-entropies against model sizes. Note that model sizes are expressed in logarithmic scale. The measurement points on each curve correspond to different parameter values. Corresponding perplexities range from 7 900 to 48 000.*

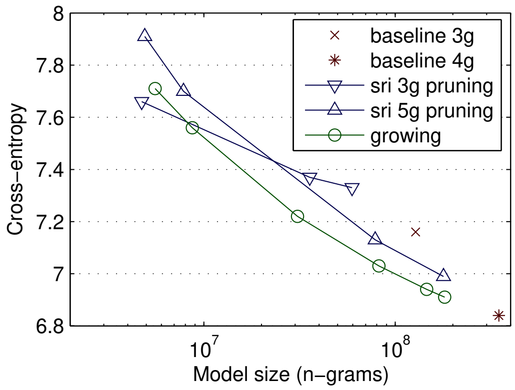

Figure 2: *English cross-entropies against model sizes, for training set of 500M words. Corresponding perplexities range from 110 to 240.*

on the other hand, was originally developed for U.S. English and has been optimized for first-pass recognition using n-grams where $n < 5$. The SONIC recognizer currently does not support 5-gram decoding in the first pass.

3.3. Cross-entropy results

The baseline models were Kneser-Ney smoothed 3, 4 and 5-grams. The SRILM-toolkit [13] was used to create the entropy pruned models. For each n-gram model, we tried different values for the likelihood threshold parameters and got models of varying sizes. The tests for the growing method were run with different values of α, yielding models of different sizes. The performance of the models was measured by cross-entropy $H_{\mathcal{M}}$ of the model $\mathcal{M}$ and a previously unseen test set T:

$$H_{\mathcal{M}}(T) = -\frac{1}{W_T} \log_2 P(T|\mathcal{M}) \tag{7}$$

Cross-entropy is directly related to perplexity, as $ppl(T) = 2^{H(T)}$, but the relative changes in cross-entropy reflect the relative changes in word error rate better.

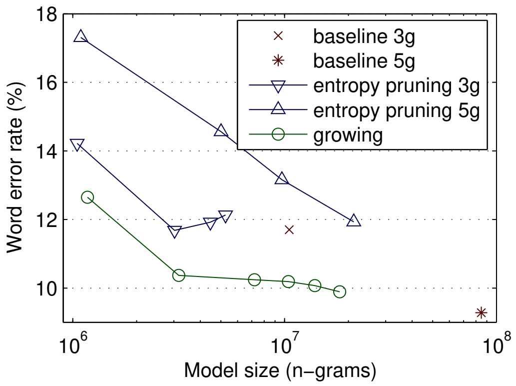

Figure 3: *Word error rate against model sizes for Finnish. The corresponding phoneme error rate ranges from 1.4 to 2.4.*

The cross-entropy results for Finnish data are plotted in Figure 1. With the English data, we tested how the methods work with different amounts of training data, ranging from 5M words to 500M words. The results for the biggest training set are plotted in Figure 2. The amount of training data changed the relative performances of the model so that the smaller the data set, the less helpful it was to have higher order n-grams in the model. The cross-entropy for the best models with 10^7 n-grams changed approximately from 8.3 to 7.5 with the training set size.

From the Finnish results we can see that entropy pruning from trigrams gives better small models than entropy pruning from 5-grams. It is clear that this entropy pruning is not optimal, since pruning from 5-grams should always give better model than pruning from trigrams. The growing method outperforms the entropy pruning, but since we use greedy sampling, it is not optimal either.

The difference in cross-entropies of Finnish and English language models is not surprising. The entropy of Finnish language is naturally higher, since one word can contain information of several English words (e.g. "from the pub also", "kuppilastakin"). It is interesting to see that the entropy pruned 5-grams seem to work relatively better for English material than for Finnish.

3.4. Speech recognition results

The Finnish speech recognition experiments were run on a subset of models used in the cross-entropy experiments. The results are plotted in Figure 3. For the English experiments, we had to limit the highest order n-gram to 4 due to the current n-gram length limitations of SONIC for first-pass decoding. We were unable to test the largest language models due to restrictions related to 32-bit memory space access (i.e., models greater than 2GB in size). The results are plotted in Figure 4.

In the Finnish speech recognition experiments, the proposed method performs relatively better than in the cross-entropy experiments. It outperforms the other models for all model sizes. The results of the English speech recognition experiments were somewhat surprising. The entropy pruned trigram models were slightly better than the models trained by the growing method and the improvements predicted by the cross-entropy experiments were not achieved.

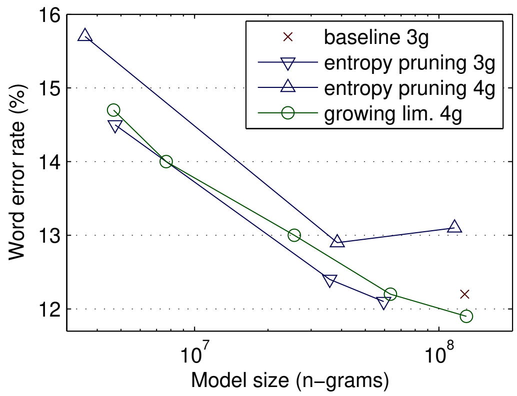

Figure 4: *Speech recognition results for English. The models were limited so that no model contained higher order grams than 4-grams. The largest models were not tested.*

4. Discussion and conclusions

We presented a method for building an n-gram model incrementally. In contrast to the pruning methods, the growing algorithm never handles the full unpruned model. This allows us to consider higher order n-grams for inclusion into the model and also leads to smaller memory consumption during model estimation.

The proposed growing method was compared to entropy based pruning [1]. Neither method is optimal. Both methods use a greedy search to find the best model. Neither of the algorithms takes into account that in speech recognition experiments, the word history might contain errors. In this case, it is more useful to keep a low order n-gram in the language model than a high order n-gram. The lower order n-gram should also generalize better for new sentences.

In the cross-entropy experiments in both Finnish and English, the growing method outperforms the other methods. The small models created by the growing method performed relatively better than the ones in our previous experiments [2]. This difference is due to the final pruning step, which was not carried out in the older experiments. The relative performance of the models is quite different for Finnish and English. In the Finnish material, words were split into smaller units (26k different units) whereas in English, 64k most common words were used. The longer span dependencies are probably more important for the Finnish language models. This difference in training data structure probably affected the tested algorithms differently and explains the changes in relative performance.

In the Finnish speech recognition experiments, the proposed growing method outperformed the entropy pruned models. In light of this and earlier results it is surprising that the entropy pruned trigram model gave the best results with respect to model size in the English experiments. The growing method with the 4-gram limit gave the best absolute performance and if the larger models had been tested, the difference would probably have been bigger. As for the reasons, we can only speculate that for English it is probably not as helpful to use higher order n-grams than for Finnish. It is also possible that tuning the pruning parameters of the SONIC system for 4-grams could affect the results.

5. Acknowledgments

We would like to thank Finnish news agency (STT) and Finnish IT center for science for providing text data. The Finnish Federation of the Visually Impaired and Departments of Speech Science and General Linguistics of the University of Helsinki are thanked for the Finnish audio data. We thank Mathias Creutz for the discussion leading to development of this model. The rest of the NNRC speech recognition group is thanked for providing their tools and scripts for the Finnish experiments.

6. References

[1] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[2] V. Siivola, "Building compact language models incrementally," in *Proceedings of Second Baltic Conference on Human Language Technologies*, 2005, pp. 183–188.

[3] J. Rissanen, *Stochastic complexity in statistical inquiry theory*. World Scientific Publishing Co., Inc., 1989.

[4] E. Whittaker and B. Raj, "Quantization-based language model compression," in *Proc. of Eurospeech*, 2001, pp. 33–36.

[5] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech and Language*, vol. 13, no. 4, pp. 359–393, 1999.

[6] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, Eds., *Numerical recipes in C*, 2nd ed. Cambridge University Press, 1997.

[7] "Finnish Text Collection," 2005, collection of Finnish text documents from years 1990–2000. Compiled by Department of General Linguistics, University of Helsinki, Linguistics and Language Technology Department, University of Joensuu, Research Institute for the Languages of Finland, and CSC. Available at http://www.csc.fi/kielipankki/

[8] V. Siivola, T. Hirsimäki, M. Creutz, and M. Kurimo, "Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner," in *Proc. of Eurospeech*, 2003, pp. 2293–2296.

[9] T. Hirsimäki, M. Creutz, V. Siivola, and M. Kurimo, "Morphologically motivated language models in speech recognition," in *Proceedings of International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, 2005.

[10] J. Pylkkönen, "An efficient one-pass decoder for finnish large vocabulary continuous speech recognition," in *Proceedings of Second Baltic Conference on Human Language Technologies*, 2005, pp. 167–172.

[11] B. Pellom, "Sonic: The University of Colorado continuous speech recognizer," University of Colorado," Technical Report TR-CSLR-2001-01, 2001.

[12] B. Pellom and K. Hacioglu, "Recent improvements in the CU SONIC ASR system for noisy speech: The SPINE task," in *Proc. of ICASSP*, 2003, pp. I–4 – I–7.

[13] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proc. of ICSLP*, 2002, pp. 901–904, http://www.speech.sri.com/projects/srilm/.