Quantization-based Language Model Compression

E. W. D. Whittaker and B. Raj

Compaq Cambridge Research Laboratory
Cambridge, MA 02142 USA

Abstract

This paper describes two techniques for reducing the size of statistical back-off N-gram language models in computer memory. Language model compression is achieved through a combination of quantizing language model probabilities and back-off weights and the pruning of parameters that are determined to be unnecessary after quantization. The recognition performance of the original and compressed language models is evaluated across three different language models and two different recognition tasks. The results show that the language models can be compressed by up to 60% of their original size with no significant loss in recognition performance. Moreover, the techniques that are described provide a principled method with which to compress language models further while minimising degradation in recognition performance.

1. Introduction

In this paper we investigate the effect that quantizing language model (LM) parameters has on the size and recognition performance of several statistical N-gram back-off language models. We describe two methods for compressing language models each of which involves quantizing language model probabilities and back-off weights and includes a stage of parameter pruning. The aim of the techniques is simple: to reduce the size of the language model in memory while minimising any degradation in recognition performance. There are several compelling reasons for addressing this issue. The main reason is that the language model is in general by far the largest component of a speech recognition system. From desktop dictation applications to incorporating speech on hand-held PCs, memory limits the size of the language model that can be used and severely restricts the performance of the speech recognition system.

Several techniques have been described in patents and the language modelling literature on methods for minimising the size of language models in memory. One method described in [1] avoids the issue to some extent by storing the language model on disk and reading the necessary portions from disk as they are required. Effective caching routines make this an effective solution although it is clearly slower than storing the language model in memory and is unsuitable for speech applications on hand-held devices. Other methods have concentrated on compressing the word identifiers and manipulating floating point parameter values [2]. We believe that no one has reported results on quantizing language model parameters in the way we describe in this paper.

We begin by briefly describing the conventional storage structure of a back-off language model in Section 2. In Section 3 we describe the application of quantization to compression. In Section 4 we describe the first compression method, which we call *absolute parameter compression* which quantizes language model probabilities and back-off weights directly and in Section 5 we describe the second method, which we refer to as *difference parameter compression* which differs from the first by quantizing the differences between language model probabilities and their backed-off probabilities. In Section 6 we describe the two recognition tasks that are used to evaluate the effect of quantization, the composition and performance of the three baseline language models and the results of the compression methods on each language model. We discuss the findings and offer conclusions in Sections 7 and 8.

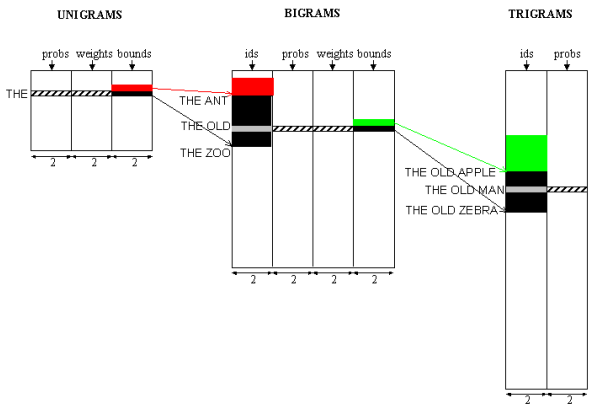

Figure 1: Tree structure storage of a back-off word trigram language model showing the search for the trigram: THE OLD MAN. Each unigram entry requires 6 bytes of storage, each bigram entry requires 8 bytes and each trigram entry requires 4 bytes.

2. Language model storage

There are many ways in which back-off N-gram language models can be stored in practice, however using hash tables and storing N-gram counts or probabilities in a tree structure are probably the most common methods. In this paper, we assume that N-gram *probabilities* are stored in a tree structure as shown in Figure 1.

The tree structure originates from a hypothetical root node (not shown) which branches out into the unigram nodes at the first level of the tree, each of which branches out to bigram nodes at the second level and so on. Each node in the tree has a word id associated with it which represents the N-gram for that word with a context represented by the sequence of words from the root of the tree up to (but not including) the node itself. For vocabularies with less than 65536 words, the word ids generally use 2-byte representations. In addition, all nodes have

a probability associated them and all non-terminal nodes have an associated back-off weight. Rather than store 4-byte floating point values for probabilities and back-off weights the values are normally quantized by truncating them so that the number of unique values is less than 65536. The 4-byte values are stored in a look-up table and 2-byte indices into the appropriate probability and back-off weight look-up tables are stored in nodes instead of the values themselves. The information for all nodes at a particular level in the tree is stored in sequential arrays as shown in Figure 1. Each array in the ith level of the tree represents sequential blocks of child nodes of the parent nodes in the $(i-1)$th level of the tree. The largest index of each block is the *boundary* value for the block which is stored in the parent node of that block. Since blocks are consecutive the boundary value of a parent node in the $(i-1)$th level together with the boundary value of the (sequentially) previous parent node at the same level specifies the exact location of the children of that node at the ith level. To locate a specific child node a binary search of word ids is performed between the two specified boundary values. In Figure 1, the example of searching for $P(\text{MAN}|\text{THE, OLD})$ is given. Firstly, the word id of THE is determined. The upper boundary value of THE's children nodes at the bigram level is given by the value in the boundary array (bounds) associated with the word id of THE. The boundary value specified by the word before THE in the word id array (ids), plus one, specifies the lower boundary position of THE's children. A binary search of word ids is then performed between the upper and lower boundary values for the word id of OLD. A similar process is then performed at the bigram level culminating in a search of word ids of THE OLD's children at the trigram level for the word id of MAN. Finally, the position of MAN in the word ids array is also the position of $P(\text{MAN}|\text{THE, OLD})$ in the probability array (probs).

Since the boundary values can be very large, 4-byte numbers would be required. However, a 4-byte number can be represented as $A*2^{16}+B$ where $B<2^{16}$, so boundary values can be stored using 2-bytes and an additional look-up table. Each entry in the look-up table stores the index of the last node that has a boundary value whose first 2 bytes are the same number as the index of that entry in the look-up table. A binary search of values in the look-up table is performed for the $(i-1)$th and ith look-up table entries between which the value of the position of the parent node lies. The value of A is thus given by i. The value of B is stored with the parent node itself. This is the scheme employed in the CMU-Cambridge Toolkit [3] for example.

Thus, in summary, each unigram entry needs a total of 6 bytes of storage, each bigram entry needs 8 bytes, and each trigram entry needs 4 bytes. The total size of the language models is 6*$N(unigrams)$ + 8*$N(bigrams)$ + 4*$N(trigrams)$, where $N(\cdot)$ is the number of the types of events in parentheses. Memory for hashing of vocabulary entries and memory allocation overheads were not considered significant.

3. Quantization

Quantization is the process by which a variable with a continuous range of values is mapped onto one of a discrete set of quantization levels. Quantization provides an effective way of reducing the number of bits needed to store a variable, at the cost of introducing errors in its value (i.e. the difference between the true value of the variable and its quantized value). The quantization levels are stored in an index table and indices into this table are stored against the variable rather than its value. A variable quantized to 2^N quantization levels can thus be stored using only N bits. For this reason this is also referred to as N-bit quantization.

In this paper we use the Lloyd-Max algorithm [4] for the determination of optimal quantization levels. This is an iterative procedure which estimates the quantization levels that minimize the average squared error introduced in the variable by quantization. The resulting quantization levels are more densely placed in regions where the density of the variable is high and more sparsely located in regions of low density.

4. Absolute parameter compression

This method assumes that a back-off trigram language model has already been built in which for each N-gram ($N=1,2,3$) event there is an associated probability and for each context there is a back-off weight. Model compression is achieved in two steps: a) quantization of N-gram probabilities and back-off weights and b) pruning of parameters from the language model.

4.1. Quantization

All unigram, bigram and trigram probabilities and unigram and bigram back-off weights are quantized to a small number of quantization levels. Quantization is performed separately on each of the N-gram probability and back-off weight lists and separate quantization level look-up tables generated for each of these sets of parameters. Compression results from the reduced number of bits needed to store the indices into the look-up tables. We investigated quantizing parameters using $1,2,3,\ldots,8$ bits.

4.2. Pruning

Pruning is the process by which some N-gram events are discarded from the language model to reduce the size of the model. The pruning method proposed here uses the following heuristic. If $Q_i^{\{P,\alpha\}}[\cdot]$ is a function that maps either a probability (P) or back-off weight (α) in the i-gram table to its quantized value, $P(\cdot)$ is the probability of an event and $\alpha(\cdot)$ is the back-off weight of some context, then if:

$$Q_3^P[P(w_i \mid w_{i-2}, w_{i-1})] = Q_3^P[Q_2^\alpha[\alpha(w_{i-2}, w_{i-1})] \cdot Q_2^P[P(w_i \mid w_{i-1})]] \quad (1)$$

the explicitly stored trigram event (w_{i-2}, w_{i-1}, w_i) is discarded from the language model. This heuristic is similar in philosophy to the method described in [5]. Pruning was only performed on trigram events although the same principle can easily be extended to bigram events.

5. Difference parameter compression

This method also assumes that the language model has already been constructed. Once again, language model compression is achieved both by quantization and pruning.

5.1. Quantization

Here, for N-gram events where $N>1$, we quantize the difference between N-gram probabilities and their quantized backed-off estimates. The stored values now represent indices to the quantized probability differences. During recognition the true

probability must be composed by adding the backed-off estimate to the quantized differences. Unigram probabilities and all back-off weights are quantized as in Section 4.

Procedurally, first the unigram probabilities and back-off weights are quantized. Bigram back-off weights and the differences between the true bigram probabilities and their quantized backed-off estimates are then quantized. Finally the differences between the true trigram probabilities and their quantized backed-off estimates are quantized.

5.2. Pruning

To prune parameters an additional quantization level of zero value is enforced during quantization. This results in two benefits. First the additional quantization level results in lower average quantization error. Second, all parameters whose difference is quantized to zero need not be explicitly stored and can be pruned, since these parameters can now be entirely composed by backing-off with no additional loss (other than that introduced by the quantization itself).

6. Experimental work

6.1. Recognition tasks

Experiments were run to evaluate the effect of the proposed compression schemes on the size and the recognition performance of language models. Two different recognition tasks were used in the experimental work in this paper: the 1998 DARPA HUB4 evaluation [6] and the 2000 DARPA SPINE evaluation [7]. Both tasks differ significantly in difficulty and hence the performance of state-of-the-art systems. All recognition experiments use the CMU SPHINX-III recognition system [8]. For the DARPA HUB4 evaluation task continuous density 3-state HMMs with 6000 tied states, each modelled by a mixture of 16 Gaussians, were trained using the 1996 Hub4 training data provided by LDC. For the experiments on the SPINE task 3-state continuous density HMMs with 2600 tied states, each modelled by a mixture of 8 Gaussians, were used. The acoustic models and the language model for the SPINE task were provided by Carnegie Mellon University.

6.2. Baseline language models

Two different language models were investigated for the HUB4 task. The two models use different vocabularies, different discounting schemes, different training data and were built using different tools. Only one language model was used for the SPINE task. It is expected that assessing the impact of the compression techniques across tasks and language models will be more meaningful than investigating numerous language models on only one task. The performance and salient characteristics of each language model are given in Table 1.

LM	Task	# N-grams	size (Mb)	WER%
1	HUB4	15,134,669	75.7	22.0
2	HUB4	20,323,339	100	20.9
3	SPINE	45,189	0.241	33.9

Table 1: Characteristics and baseline performance of LMs.

The original language models in the subsequent experiments are identified using the LM identifier in the first column of the table.The total number of N-grams ($N = 1, 2, 3$) in each language model is shown in the third column of Table 1 and the approximate size in Mb is shown in the fourth column. The final column shows the word error rate (WER) achieved on the second evaluation set of the 1998 DARPA Hub4 evaluation for the HUB4 models and the WER achieved on the SPINE evaluation set using the SPINE language model.

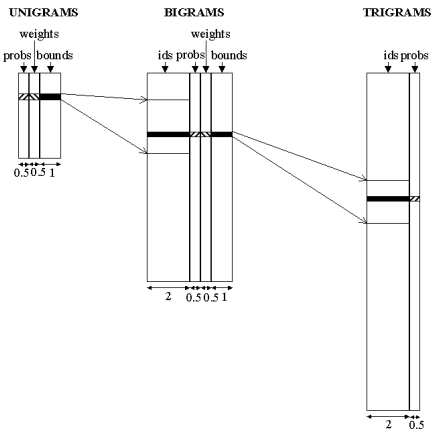

Figure 2: Tree structure storage of back-off trigram language model showing compression achieved through parameter quantization and pruning. The number of bytes required to store any event is much less than the corresponding number in Figure 1.

Figure 2 shows the storage structure for a compressed language model. This is essentially the same as the structure in Figure 1 except that all lists are narrower due to the reduced number of bits needed to store parameters. The lists are also shorter when parameters are pruned. In addition, the number of bits used to store boundary values is also reduced by representing boundary values as $A * 2^{24} + B$ where $B < 2^8$, and storing the one byte offset value B in each non-terminal node of the tree, and an additional 3-byte look-up table for A.

LM	Bits	# 3-gram dels.	size (Mb)	WER%
1	4	0	43.5	22.2
1	4	1686294	39.3	22.2
1	2	0	40.6	23.1
1	2	5260335	36.9	23.3
2	4	0	58.0	21.1
2	4	457148	56.8	21.3
2	2	0	54.1	22.7
2	2	5348062	42.1	23.2
3	4	0	0.134	34.0
3	2	0	0.127	34.3

Table 2: Recognition performance of language models quantized using absolute parameter compression.

Table 2 shows the results obtained when absolute parameter compression was used to compress the language models. The results reported are for 4-bit and 2-bit quantization of trigram probabilities. In all cases unigram and bigram probabilities and back-off weights were quantized to 4 bits. In cases where pruning was used, the number of deleted trigrams is shown in the

third column. Zero trigram deletions indicate that pruning was not attempted.

LM	Bits	# 3-gram dels.	size (Mb)	WER%
1	4	1119492	40.7	22.1
1	2	3526503	32.8	22.5
2	4	201013	57.5	21.1
2	2	932822	52.0	21.7
3	4	3379	0.126	34.1
3	2	9651	0.106	34.2

Table 3: Recognition performance of language models quantized using difference parameter compression.

Table 3 shows the results obtained when difference parameter compression was used to compress the language models. Recognition results obtained using 4-bit and 2-bit quantization of trigram probabilities are shown. In all cases unigram and bigram probabilities and back-off weights were quantized to 4 bits. The number of trigrams deleted from each language model is given in the third column.

7. Discussion

It is clear from the results that the number of quantization levels required for each set of parameters can be made remarkably small while still preserving essentially the same recognition accuracy. In addition, both compression techniques highlight unnecessary parameters that can then be discarded. As expected, the performance degradation increases as fewer quantization levels are used to represent the parameters and also as more parameters are discarded. There is a trade-off evident between the two techniques in terms of the performance for the number of operations needed to retrieve an N-gram probability. Parameters compressed using absolute parameter compression can be retrieved directly from the LM. On the other hand, difference parameter compression, while resulting in slightly lower degradation, requires that backed-off probabilities of events be retrieved in order to retrieve the probability of the event. Also, the pruning achieved using the methods described in this paper is greater for absolute parameter compression.

Contrary to most language model compression schemes that attempt to reduce LM size by length-wise compression, i.e. reduction of the length of the lists of stored events, the schemes proposed in this paper concentrate on width-wise compression, i.e. reduction in the amount of storage required for any specific event. Length-wise compression is achieved primarily as a side effect of the width-wise compression. However, the proposed compression schemes should also work equally well in conjunction with any other length-wise compression scheme, resulting in cumulative reduction of LM size. This is examined in [9].

The quantization scheme used in this paper, Lloyd-Max quantization, optimized the average quantization error in LM parameters. While this was observed to result in improved performance over linear quantization, it is expected that quantization schemes that utilize better optimization criteria such as perplexity would further reduce the degradation in performance due to parameter quantization.

Although the discussion in this paper has related to back-off word N-gram language models it is clear that the techniques are directly applicable to back-off N-gram models that use different modelling units, e.g. classes or sub-word units. Indeed, in any language model where a probability is stored for a language 'event' it is likely that the compression schemes that have been presented or variations on them could be applied with success.

8. Conclusion

In this paper we have described two techniques that compress language models while minimising the degradation in recognition performance. Language models used on large vocabulary broadcast news tasks have been compressed by up to 60% of their original size with minimal loss in recognition performance using a combination of parameter quantization and parameter pruning. We have shown that the techniques work equally well for different language models on the same recognition task as for a language model for a different task. These methods show what can be achieved in terms of parameter value compression and to some extent what can be achieved through pruning. Further compression strategies should address ways in which the tree structure itself can be compressed.

9. Acknowledgements

The authors wish to thank Stan Chen for the use of his HUB4 language model in the experiments in this paper and Rita Singh for providing us with the SPINE acoustic and language models.

10. References

[1] M.K. Ravishankar, *Efficient Algorithms for Speech Recognition*, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, 1996.

[2] D. Kanevsky and S.P. Rao, "System and Method for Providing Lossless Compression of N-gram Language Models in a Real-time Decoder," US Patent US6092038, July 2000.

[3] P. R. Clarkson and R. Rosenfeld, "Statistical Language Modeling Using the CMU-Cambridge Toolkit," in *Proceedings of the European Conference on Speech Communication and Technology*, Rhodes, Greece, 1997.

[4] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, 1982.

[5] K. Seymore and R. Rosenfeld, "Scalable Backoff Language Models," in *Proceedings of the International Conference on Spoken Language Processing*, Philadelphia, USA, 1996.

[6] D. S. Pallet, J. G. Fiscus, J. S. Garofolo, A. Martin, and M. A. Przybocki, "1998 Broadcast News benchmark Test Results," in *Proceedings of the DARPA broadcast news workshop*, Feb 28 - Mar 3 1999.

[7] R. Singh, M. L. Seltzer, B. Raj, and R. M. Stern, "Speech in Noisy Environments: Robust Automatic Segmentation. Feature Extraction and Hypothesis Combination," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Salt Lak City, Utah, May 2001.

[8] P. Placeway et al., "The 1996 HUB-4 SPHINX-3 System," in *Proceedings 1997 DARPA Speech Recognition Workshop*, Chantilly, Virginia, Feb 2-5 1997.

[9] E. W. D. Whittaker and B. Raj, "Comparison of Width-wise and Length-wise Language Model Compression," Submitted to *the European Conference on Speech Communication and Technology*, Aalborg, Denmark, 2001.