Teemu Hirsimäki. 2007. On compressing n-gram language models. In: Proceedings of the 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2007). Honolulu, Hawaii, USA. 15-20 April 2007, pages IV-949-952.

ON COMPRESSING N-GRAM LANGUAGE MODELS

Teemu Hirsimäki

Helsinki University of Technology
Adaptive Informatics Research Centre
P.O.Box 5400, 02015 HUT, Finland

ABSTRACT

In large-vocabulary speech recognition systems, the major part of memory resources is typically consumed by a large n-gram language model. Representing the language model compactly is important in recognition systems targeted for small devices with limited memory resources. This paper extends the compressed language model structure proposed earlier by Whittaker and Raj. By separating n-grams that are prefixes to longer n-grams, redundant information can be omitted. Experiments on English 4-gram models and Finnish 6-gram models show that extended structure can achieve up to 30 % lossless memory reductions when compared to baseline structure of Whittaker and Raj.

Index Terms— Data structures, Speech recognition, Natural languages, Modeling, Data compression

1. INTRODUCTION

The major part of memory consumption of large-vocabulary continuous speech recognition systems is usually due to the size of statistical language models. Especially, in general recognition tasks where the vocabulary or topic can not be restricted, the recognition accuracy can be improved by obtaining larger text corpora and training larger language models. While the memory resources are often not the main concern in research systems, consumer systems have to take the memory issues into account. Thus representing the language models efficiently affects the recognition accuracy directly on systems with limited memory resources.

Entropy pruning [1] is a widely used method for reducing the number of language model parameters. Full n-gram statistics can be reduced considerably before recognition accuracy starts to degrade. Another approach for reducing the number of parameters is to grow models incrementally [2]. Goodman and Gao [3] have shown that combining pruning and clustering can reduce the number of parameters further. Whittaker and Raj [4, 5, 6], on the other hand, have proposed several lossless and lossy compression methods for storing the language model parameters efficiently while maintaining reasonable access times. Olsen and Oria [7] have compressed 2-gram models by using codebooks for probability distributions.

This paper presents an extension to compressed data structure of Whittaker and Raj. The baseline structure and compression methods are presented in Section 2 and the extended structure in Section 3. Section 4 presents experiments with discussion.

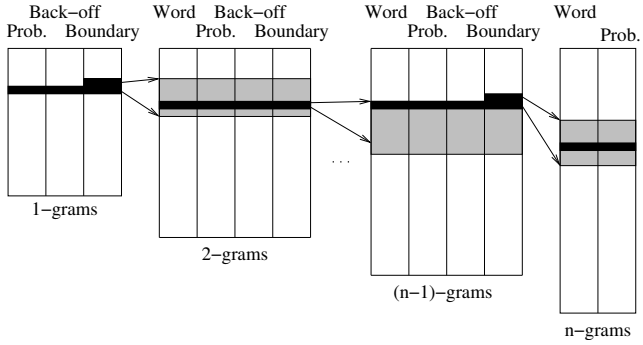

Fig. 1. The baseline tree structure.

2. BASELINE STRUCTURE

2.1. Back-off language model

In the rest of the paper, we assume that the language model is represented in a common back-off format. An back-off m-gram model $\mathcal{M}$ is a tuple $(V, G, \alpha_\mathcal{M}, \beta_\mathcal{M})$, where

- V is the set of symbols (usually whole words or sub-word units)
- $G = (G_1 \cup \cdots \cup G_m)$ is the set of n-grams stored explicitly in the model.
- $\alpha_\mathcal{M} : G \to \mathbb{R}$ is a table of log-probabilities of the n-grams stored explicitly in the model,
- $\beta_\mathcal{M} : G \to \mathbb{R}$ is a table of logarithmic back-off weights of the n-grams stored explicitly in the model.

This corresponds to the widely used back-off structure introduced by Katz [8]. Given an arbitrary n-gram $w_1^n = (w_1, \ldots, w_n)$, the conditional log-probability $\log \Pr(w_n|w_1^{n-1})$ is computed from the model recursively as follows:

$$\log \Pr(w_n|w_1^{n-1}) = \begin{cases} \alpha_\mathcal{M}(w_1^n) & \text{if } w_1^n \in G, \\ \beta_\mathcal{M}(w_1^{n-1}) + \log \Pr(w_n|w_2^{n-1}) & \text{if } w_1^{n-1} \in G, \\ \log \Pr(w_n|w_2^{n-1}) & \text{otherwise.} \end{cases} \quad (1)$$

2.2. Baseline tree structure

The tree structure used by Whittaker and Raj is illustrated in Figure 1. The n-grams of different orders are stored in separate tables, and higher-order tables are accessed through lower-order tables. A row in n'th table correspond to an n-gram $w_1^n = (w_1, \ldots w_n)$. Each table contains the following arrays:

1. The index of the word w_n. This array is not required for 1-grams, if we assume that 1-gram table contains a row for each word in vocabulary
2. The log-probability $\alpha_{\mathcal{M}}(w_1^n)$.
3. The back-off weight $\beta_{\mathcal{M}}(w_1^n)$. By the definition of the back-off model, this field is not required at the highest-order.
4. A boundary index. It indicates where is the last child of the row in the next table. More specifically, it is first row (in the next table) after $(n+1)$-grams that have w_1^n as prefix, i.e., n-grams $(w_1, \ldots, w_n, \cdot)$. Naturally, the boundary values can be omitted from the highest-order table.

For example, imagine we want to fetch $\alpha_{\mathcal{M}}(\textit{in}, \textit{front}, \textit{of}, \textit{the})$ from the structure shown in Figure 1. We start at the first table, and move to row r corresponding to word *in* (the black row in the figure). Boundary values of rows r and $r-1$ indicate the range for 2-grams $(\textit{in}, \cdot)$ in the second table. In that range, we search the row corresponding to word *front*. Assuming that the rows of the range are sorted according to the word indices, the desired row can be found efficiently by binary search. Again we look at the boundary values and proceed further through the third table to the fourth table. Finally, the probability field of the row corresponding to the word *the* gives $\alpha_{\mathcal{M}}(\textit{in}, \textit{front}, \textit{of}, \textit{the})$. The main benefit of the above structure is that branches can be represented very compactly—by a single value per parent node.

2.3. Compressing the fields

In each array, every row has the same bit-width to allow for constant-time access to an arbitrary row. For each array, we can choose the minimum number of bits that can represent all values. For example, if word indices varied between 0 and 8 428, only 14 bits would be required for the word index array ($\log_2 8428 \approx 13.04$).

Probability values and back-off weights are usually represented as 32-bit floating point values in computer systems. However, 32 bits is more than enough for storing parameters of typical n-gram language models. Whittaker and Raj showed that the probabilities and back-off weights can be quantized even down to 8 bits before the quantization degrades speech recognition results [5].

Whittaker and Raj also described a general method for compressing sorted integer arrays [6], which can be used to compress boundary values and word indices. In the original paper, the computational overhead of the integer compression was not evaluated, but the access time should be roughly logarithmic with respect to length of the array. Without the compression the access time would naturally be constant. In this paper, the boundary array compression is used, but the word index compression is omitted in order to keep the structures and experiments simpler. The effect of the omission is discussed in more detail in Section 4.2.

3. EXTENDED STRUCTURE

In the baseline structure, the back-off weight and boundary value arrays can be omitted from the highest order because there are no higher-order children. However, there are also childless rows on the lower orders, but we have to store a back-off weight (log-zero) and boundary value (same as the previous boundary value) anyway. This overhead becomes especially evident with entropy pruned models, since typically high-order n-grams get pruned more easily, and many childless n-grams are left at the lower orders.

Instead of storing two values (the back-off weight and boundary value) for every row, it may be more efficient to store a single pointer value for every row, and omit childless rows from the back-off and boundary arrays as shown in Figure 2. The pointer value simply tells the index to use for back-off and boundary arrays. If row r is childless, the pointer value $\mathrm{ptr}(r) = \mathrm{ptr}(r-1)$. Otherwise, the pointer value $\mathrm{ptr}(r) = \mathrm{ptr}(r-1) + 1$. Since the pointer arrays are always sorted, they can be compressed in the same way as boundary arrays using Whittaker and Raj's integer compression. The reader may notice that the pointer array could be represented by one bit per row (indicating whether the value is greater than previous). However, retrieving the values from the array would then be unreasonably slow: linear with respect to length of the array.

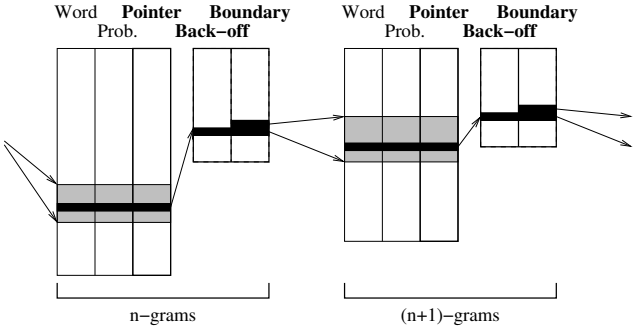

Fig. 2. Extended structure for orders n and $n+1$. The arrays of each order have been split into two parts (modified arrays shown in bold). Back-off and boundary values are stored only for rows that have children, but pointers are needed to access them.

For some orders, the extended structure may actually consume more memory, in which case the order is stored in the baseline format. If integer compression is used, the amount of compression achieved by the extended structure is hard to predict without building the structure, but experimental results presented in Section 4 shed light on the issue. Also computational issues are discussed there.

4. EXPERIMENTS

4.1. Setup

The efficiency of the extended structure was evaluated on English and Finnish language models.

The English models were trained using the New York Times partition of the English Gigaword corpus [9]. 927 million words were used for training, and 197 thousand words for computing perplexities. The vocabulary was limited to 50000 words, and a 4-gram model was trained with Good-Turing smoothing.

The Finnish models were trained on the Kielipankki corpus, available from CSC [10]. The training set contained 145 million words from newspapers, magazines and books. The evaluation set contained 149 thousand words. Before training the language models, the words were split into 8428 sub-word units using the Morfessor algorithm [11]. Splitting the words into smaller units improves the language modeling performance considerably in Finnish and can avoid the problem of out-of-vocabulary words [12]. A 6-gram model was trained with Good-Turing smoothing.

For both languages, entropy pruning was used for producing smaller models with pruning thresholds $10^{-9}, 10^{-8}$ and 10^{-7}. The SRI Language Modeling Toolkit [13] was used for training and pruning. The Finnish models were trained with the default cut-off values: all n-grams occurring only once were ignored for $n > 2$. In the English case, 4-grams occurring only twice were also ignored.

Pruning	Baseline (MB)	Extended (MB)	Saving (%)	Compression / Float bits
10^{-7}	14.3	11.9	17.2	yes / 8
10^{-8}	128.5	116.6	9.2	yes / 8
10^{-9}	463.3	418.9	9.6	yes / 8
none	637.8	582.0	8.7	yes / 8
10^{-7}	20.7	19.1	7.8	no / 8
10^{-8}	183.4	183.4	0.0	no / 8
10^{-9}	666.7	666.7	0.0	no / 8
none	920.6	920.6	0.0	no / 8
10^{-7}	20.7	15.8	23.7	yes / 16
10^{-8}	182.5	156.3	14.4	yes / 16
10^{-9}	654.8	558.9	14.6	yes / 16
none	899.5	776.5	13.7	yes / 16
10^{-7}	27.1	23.1	15.0	no / 16
10^{-8}	237.5	229.5	3.4	no / 16
10^{-9}	858.2	829.5	3.3	no / 16
none	1182.4	1150.8	2.7	no / 16
10^{-7}	33.5	23.7	29.2	yes / 32
10^{-8}	290.7	235.5	19.0	yes / 32
10^{-9}	1037.7	838.8	19.2	yes / 32
none	1423.0	1165.5	18.1	yes / 32
10^{-7}	39.9	31.0	22.4	no / 32
10^{-8}	345.7	310.9	10.1	no / 32
10^{-9}	1241.2	1109.4	10.6	no / 32
none	1705.8	1539.8	9.7	no / 32

Table 1. English models: Comparing the sizes of the baseline and extended structures, and the relative saving obtained by using extended structure instead of the baseline structure.

Pruning	Baseline (MB)	Extended (MB)	Saving (%)	Compression / Float bits
10^{-7}	8.8	7.2	18.3	yes / 8
10^{-8}	52.4	44.8	14.4	yes / 8
10^{-9}	201.8	190.3	5.7	yes / 8
none	407.3	407.0	0.1	yes / 8
10^{-7}	12.8	11.9	6.9	no / 8
10^{-8}	78.8	76.2	3.3	no / 8
10^{-9}	303.2	303.2	0.0	no / 8
none	580.5	580.5	0.0	no / 8
10^{-7}	13.0	9.8	24.8	yes / 16
10^{-8}	76.8	60.9	20.6	yes / 16
10^{-9}	293.2	261.7	10.8	yes / 16
none	582.4	581.5	0.1	yes / 16
10^{-7}	17.0	14.6	14.2	no / 16
10^{-8}	103.2	94.2	8.7	no / 16
10^{-9}	394.5	387.3	1.8	no / 16
none	755.6	755.6	0.0	no / 16
10^{-7}	21.5	15.0	30.1	yes / 32
10^{-8}	125.6	93.2	25.8	yes / 32
10^{-9}	475.9	402.3	15.5	yes / 32
none	932.6	904.7	3.0	yes / 32
10^{-7}	25.5	19.8	22.2	no / 32
10^{-8}	152.0	126.8	16.6	no / 32
10^{-9}	577.3	543.8	5.8	no / 32
none	1105.8	1105.4	0.0	no / 32

Table 2. Finnish models: Comparing the sizes of the baseline and extended structures, and the relative saving obtained by using extended structure instead of the baseline structure.

The compression performance of the structures was evaluated by computing the memory footprint depending on the entropy pruning threshold, quantization level of the floating point values, and whether integer array compression was applied. English results are shown in Table 1, and Finnish results in Table 2.

The computational efficiency of the structures was evaluated by measuring the CPU time required for computing the perplexity of the test data. The perplexity computation was repeated 20 times to get robust measurements. Table 3 shows the results. The perplexities of the models and the number of n-grams are shown in Table 4.

The C++ source code for the software used in the experiments is available for download at *http://www.cis.hut.fi/thirsima/*. The package contains tools and libraries for compressing n-gram models in baseline and extended structures, and using them in applications.

4.2. Discussion

A few clear trends can be seen in Tables 1 and 2. Firstly, the more entropy pruning is applied, the more relative savings can be achieved by the extended structure. This trend is quite natural since entropy pruning creates more childless n-grams on lower orders, and the extended structure specifically tries to represent childless n-grams compactly. However, there are some differences between the languages. In the Finnish case, the extended structure does not seem to achieve any compression on the unpruned models, but in the English case it does. The reason is that higher cut-off values were used in training English models which corresponds to moderate initial pruning on the highest orders. Without the higher cut-offs, training English 4-gram models would have taken too much memory.

The second trend is that saving decreases when less bits are used for floating point values, i.e., probabilities and back-off weights. The effect of this quantization is twofold. The less bits are used for back-off weight values, the less saving the extended structure can achieve by omitting redundant back-off weights $\beta_{\mathcal{M}}$ from the childless n-grams. On the other hand, the less bits are used for the probabilities $\alpha_{\mathcal{M}}$, the smaller are the models overall, and the greater is the relative benefit from the extended structure, because probabilities are needed for all n-grams in both structures.

The third trend is that integer compression increases the relative compression of the extended structure. This is largely because the pointer arrays are sorted arrays whose values grow slowly, and the general integer compression seems to be very efficient on such arrays.

As mentioned in Section 2.3, the word index compression was not used in the experiments to keep the structures simpler. Since the extended structure deals only with back-off and boundary arrays, the word indices could be compressed normally. That would lead to smaller models overall, and the extended structure would give relatively better compression ratios.

In Table 3 we can see that the computational overhead of the extended structure depends on whether integer compression is used or not. Without compression, the additional pointer array introduces minimal overhead, but after compression, the overhead becomes visible, since accessing the compressed pointer array is roughly logarithmic with respect to the length of the array. It must be noted that the straightforward implementations could certainly be optimized further for both compressed and uncompressed structures, so the computational comparison is only suggestive. In speech recognition systems, the access to the main n-gram language model is usually

Pruning	Baseline (s)	Extended (s)	Overhead (%)	Compression
10^{-7}	85.37	102.62	20.21	yes
10^{-8}	103.48	143.88	39.04	yes
10^{-9}	133.69	185.29	38.60	yes
none	171.55	248.49	44.85	yes
10^{-7}	28.41	30.16	6.16	no
10^{-8}	36.55	38.45	5.20	no
10^{-9}	44.08	44.47	0.88	no
none	47.15	47.81	1.40	no

Table 3. Computational cost of the structures when computing perplexity of Finnish models with 32 bits per float.

	English		Finnish	
Pruning	Perplexity	n-grams (M)	Word-perplexity	n-grams (M)
10^{-7}	303	3.3	23 220	2.1
10^{-8}	232	31.1	12 805	12.5
10^{-9}	214	112.4	11 017	50.3
none	212	156.7	10 984	108.2

Table 4. Perplexities and the number of n-grams in the models. The perplexities of the Finnish models are normalized by the number of *whole words* to allow comparisons with other experiments using possibly other sub-word units for language modeling. The corresponding token perplexities are 26.3, 21.7, 20.6, 20.6.

not the most computationally expensive part of the process, so the access times of the compressed structures should be reasonable. As look-ahead language models the compressed structures are probably not applicable, since they are often accessed very heavily.

Table 4 shows the number of n-grams in the models and the perplexities on the test set. As usual, the Finnish perplexities are computed by normalizing the inverse probability with the number of *whole words* even if the model uses sub-word units as symbols. Using this "word-perplexity" makes it possible to compare perplexities with other experiments that may use different sub-word units for language modeling. The perplexities are naturally large when compared to English. A single Finnish word often carries the information of several English words by compounding words and using prefixes, suffixes, and inflections.

The order of the language models were chosen as high as possible with without requiring to use cut-offs or other special techniques heavily. In the Finnish, however, it is expected that even higher-order n-grams are useful, when the words are split into smaller units [2]. It is also probable, that the compression ratio of the extended structure gets better on higher-order models.

5. CONCLUSION

An extension to a previously proposed method for compressing language model structure was presented. By separating n-grams that are prefixes to longer n-grams, the language models can be represented more compactly without losing modeling accuracy. In experiments on English 4-gram models and Finnish 6-gram models, the extended structure obtained compression ratios between 0–30 % depending on the entropy pruning threshold, amount of quantization and integer array compression used in the original models. The best ratios are obtained on models that have been pruned with entropy pruning.

6. REFERENCES

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

[2] Vesa Siivola and Bryan Pellom, "Growing an n-gram model," in *Proceedings of the 9th European Conference on Speech Communication and Technology (Interspeech)*, Lisboa, Portugal, Sept. 2005, pp. 1309–1312.

[3] Joshua Goodman and Jianfeng Gao, "Language model size reduction by pruning and clustering," in *Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP)*, 2000, pp. 110–113.

[4] E.W.D. Whittaker and B. Raj, "Quantization-based language model compression," in *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*, 2001, pp. 33–36.

[5] E.W.D. Whittaker and B. Raj, "Comparison of width-wise and length-wise language model compression," in *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*, 2001, pp. 733–736.

[6] B. Raj and E.W.D. Whittaker, "Lossless compression of language model structure and word identifiers," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003, pp. 388–391.

[7] J. Olsen and D. Oria, "Profile based compression of n-gram language models," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2006, vol. 1, pp. 1041–1044.

[8] Slava M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech regonizer," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, Mar. 1987.

[9] David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda, "English Gigaword Second Edition," Linguistic Data Consortium, Philadelphia, 2005.

[10] "Finnish Text Collection," 2004, 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.

[11] Mathias Creutz and Krista Lagus, "Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0," Publications in Computer and Information Science A81, Helsinki University of Technology, Mar. 2005.

[12] Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen, "Unlimited vocabulary speech recognition with morph language models applied to finnish," *Computer Speech and Language*, vol. 20, no. 4, pp. 515–541, Oct. 2006.

[13] Andreas Stolcke, "SRILM – an extensible language modeling toolkit," in *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP)*, 2002, pp. 901–904, `http://www.speech.sri.com/projects/srilm/`.