PROFILE BASED COMPRESSION OF N-GRAM LANGUAGE MODELS

Jesper Olsen and Daniela Oria

Multimedia Technologies Laboratory
Nokia Research Center
Helsinki, Finland

jesper.olsen@nokia.com, daniela.oria@nokia.com

ABSTRACT

A profile based technique for compression of n-gram language models is presented. The technique is intended to be used in combination with existing techniques for size reduction of n-gram language models such as pruning, quantisation and word class modelling. The technique is here evaluated on a large vocabulary embedded dictation task. When used in combination with quantisation, the technique can reduce the memory needed for storing probabilities by a factor of 10 or more with only a small degradation in word accuracy. The structure of the language model is well suited for "best-first" type decoding styles, and is here used for guiding an isolated word recogniser by predicting likely continuations at word boundaries.

Keywords: n-gram language models, dictation, embedded systems.

1. INTRODUCTION

Speech recognition of large vocabulary natural language utterances requires an accurate language model (LM) to guide the recogniser. The dominant language model for this type of task has for the past several decades been the N-gram language model [1]. The N-gram model is conceptually a simple statistical model, which has the advantage that it can be trained automatically from data without the need to explicitly define syntactic rules for the language, or encode domain information for the task it is used in. The main disadvantages are that it typically requires large amounts of domain specific data – often millions of words – to construct a good model, and that the size of the resulting model tends to be very big. The language model will often be the largest component in the recogniser, and the size of it is therefore important: in real-time applications, a compromise always has to be made between the size of the language model that can be deployed, and the cost of the hardware which is hosting the application.

2. BACKGROUND

N-gram language models model language as a Markov source of order n-1: the probability of a word depends only on the past n-1 words. For high-end dictation systems n=3 (trigram) or higher is a common choice, whereas for more low-end systems bigrams (n=2) or even unigrams (n=1) are used. An n-gram model represents a probability distribution over all n-1 length word sequences. This means that the size (number of n-grams) in a unigram model is the same as the size of the vocabulary, V. The size of higher order n-gram models is significantly larger, a bigram model is potentially of size V*V and a trigram model of size V*V*V. In practice it is not possible to observe and train probabilities for all n-gram sequences even when a very large training corpus is used. Consequently it is common to use a "backoff" scheme [3], to calculate the probability of hypothesised word sequences which do not occur in the model by backing off to a lower order n-gram model.

Several techniques have in the past been used for reducing the size of n-gram models – most commonly these fall into the categories pruning, quantisation and word class modelling.

5.1. Pruning

The size of an n-gram model tends to grow proportionally with the size of the training corpus. Hence, one way to constrain the size of the model is to limit the size of the training corpus [4]. However, this approach makes poor use of the training data, and a better approach is to train a model on all the data and afterwards prune the model by removing n-grams. There are several ad hoc ways of doing this, e.g. removing n-grams with low frequency counts [3], or removing n-grams where backing off to a lower order n-gram gives little loss in probability [4]. Probably the most theoretically well founded approach is entropy based pruning [5] which is based on removing the n-grams in the model which results in the lowest increase in perplexity on the training data.

5.1. Quantisation

Each N-gram probability has to be represented in memory with a certain number of bits (e.g. 32 bit C-floats). The idea behind quantisation is that probabilities are stored as indexes to a codebook, and that the number of bits required for the index is less than the number of bits for the codebook probabilities. Basically all n-gram models are quantised from the beginning, because they are estimated based on frequency counts, and since many word sequences by chance have the same frequency, the number of unique probabilities is significantly smaller than the total number of n-gram probabilities in the model. But quantisation can be taken further than this by deliberately quantising probabilities using smaller codebooks. Codebooks with down to 32 elements (4-bit indexes) have been used with little degradation [6], [7].

5.1. Word Class Modelling

Introduction of word classes can be used to reduce the size of the vocabulary of the language model [8], and/or to improve the probability estimates when training data is sparse – which it often is. A moderate size reduction of the vocabulary will typically have a much larger effect in the bigram and trigram sections of the language model. Word classes can either be handcrafted to fit the dictation task, or they can be derived using statistical clustering techniques [9], which automatically discover relationships between words. Handcrafting is typically only realistic for a smaller number of word classes, but it has the advantage that the classes are better understood (e.g. name classes: person, city, company, or number classes: telephone, flight, money) and therefore are easier to adapt to fit a specific task, user or location – this can even be done dynamically if a changing "topic" can be identified [10].

3. N-GRAM PROFILES

The idea behind n-gram profiles is to order the n-grams with the same history according to probability so that the most probable is stored in position one, the second most probable in position two etc. For example, the following is a profile for bigrams that have the word YOUR in their history:

```
-0.857508        YOUR MESSAGE
-1.263640        YOUR OFFICE
-1.372151        YOUR ACCOUNT
-1.372151        YOUR HOME
-1.372151        YOUR JOB
-1.372151        YOUR NOSE
-1.372151        YOUR OLD
-1.517140        YOUR LOCAL
-1.736344        YOUR HEAD
-2.200477        YOUR AFTERNOON
```

Storing n-grams as profiles can in itself lead to memory reduction, because some words – particularly when quantisation is used – will have identical profiles or have profiles that are prefixes of longer profiles. Hence, the same profile can be reused for different sets of n-grams (the word pairs still need to be stored separately). The profiles themselves can be stored in compressed form. This can be done by "sampling" the profile, and interpolating the probabilities that are not directly represented in the profile. The simplest sampling approach is to use linear sampling – for instance by including every second sample in the profile – and calculating the value of the missing samples from their neighbours. However, it is a characteristic of the n-gram profiles that the rate of change is greater at the beginning of the profile than at the end, and a better approach is therefore to use some kind of "logarithmic" sampling, e.g.

$$i_0 = 0$$
$$i_k = i_{k-1} + \max(1,\text{round}(\log_{10}(i_{k-1})/S)) \quad (1)$$

where the scale factor, S, is used to control the spacing of the sample indexes i_k. which go into the compressed profile.

Regardless of what sampling algorithm is used, the compressed profile can be expanded into an uncompressed profile by using the stored samples to calculate the missing ones – e.g. by linear interpolation it is possible to calculate the samples $s_1,...,s_n$, between sample k and sample $k+1$ in the compressed profile (z=0,...,n):

$$s_z=p_k+z(p_{k+1}-p_k)/n \quad (2)$$

4. STORAGE STRUCTURE

In this study, the language model is used in a scenario, where it must be computationally cheap to predict word continuations from a given word. This is achieved by storing the language model in a structure containing the following components:

1. A VQ codebook with Q elements – the elements are quantised logarithmic probabilities.
2. A profile codebook with P elements – each profile is an array of indexes to the VQ codebook.
3. A unigram section consisting of two arrays where each element is an index to the VQ codebook. The first holds the unigram probability of each word, and the second the backoff probability for "forgetting" that word history in a bigram context [2].
4. A bigram or word-pair section which describes which words are allowed to follow any other. This section has V elements – one for each vocabulary word – and each element holds an index to the profile codebook, and an array which contains the word identities of the words that are allowed to follow this particular word (bigram histories). The words are ordered according to likelihood.

5. EXPERIMENTAL SETUP

Language model compression is here evaluated in the context of isolated word dictation (short pause required between words when dictating). 25 speakers are represented in the test set with 240 test utterances in addition to a short enrollment session, which is used for acoustic model adaptation (1240 biphone HMMs, 2k Gaussian mixtures). The n-gram language model used in this study was trained on the basis of a 2 million word text corpus. The language model is a bigram LM containing 23k unigrams and 255k bigrams. The bigram perplexity of the test set is 75 using the uncompressed language model.

Language model profile compression was evaluated in combination with probability quantisation. Five different quantisation levels were used: No quantisation, and quantisation using a 32, 16, 8 and a 4 element VQ codebook (corresponding to 5, 4, 3 and 2 bit indexes). In combination with this, profile compression with 5 different compression scales, S, was used (eq. 1): S=1000, 0.5, 0.1, 0.05, 0.01. The S=1000 profile compression scale is loss less for the LM used in these experiments. To illustrate these sampling scales, the five first samples selected to represent a profile are:

S=1000	1	2	3	4	5
S=0.50	1	2	3	5	8
S=0.10	1	2	9	31	65
S=0.05	1	2	16	71	156
S=0.01	1	2	71	497	1118

Table 1: first few indexes selected for profile sampling.

5.1. Memory Requirements

The same storage structure is used regardless of whether quantisation and profile compression is used. As described above in section 4, the language model has four parts: VQ Codebook, profile codebook, unigram section and word-pair section. The word-pair section is the same for all the models used here. On average each word in the vocabulary has NF=11.1 bigram continuations. The total memory required for the word-pair section is 603kByte. The VQ codebook is relatively small, between 16 and 128 byte for the four quantised model sets, and 47.6 kByte for the baseline model set – which turns out to have only 11915 unique probability values. The unigram section here is of size 184 kByte – relatively large due to the fact that these probabilities were not quantised in this study. That finally leaves the profile codebook, which is the part of the structure which is directly affected by quantisation and compression. The profile holds P profile arrays, and the size of the structure can be calculated as

$$P*4+\sum_{i=1}^{P} len(p_i)*sizeof(VQIDX) \qquad (3)$$

For the baseline LM, it is necessary to use 16-bit indexes to the VQ codebook (VQIDX). For the quantised LMs, 8-bit indexes (or less) can be used.

4. RESULTS

Table 2 below shows the word accuracy for recognition tests with each of the model sets used in this experiment. Additionally the following statistics for the profile codebook are given: number of profiles, average profile length and finally size of the profile codebook in kBytes - calculated using eq. 3, and assuming 8-bit indexes for all the quantised codebooks.

The profile statistics in table 1 is illustrated graphically in figure 1, 2 and 3. Figure one shows the number of profiles in the codebooks, figure 2 the average profile length, and finally figure 3 the size of the profile codebook (number of elements rather than kByte).

	S=1000	S=0.5	S=0.1	S=0.05	S=0.01
No Q	4033/52 436kb 85.1%	4017/10 103kb 85.0%	3849/4 51kb 84.6%	3814/3 45kb 84.0%	3810/2 39kb 79.5%
Q32	3108/64 213kb 85.0%	2797/13 48kb 85.0%	1279/6 13kb 84.6%	905/5 8kb 84.1%	456/3 4kb 81.2%
Q16	2454/78 202kb 84.9%	1773/16 36kb 85.0%	566/8 7kb 84.5%	344/6 4kb 84.1%	151/4 1kb 81.1%
Q8	937/164 157kb 73.7%	479/30 17kb 84.7%	154/12 2.6kb 84.2%	90/9 1kb 83.7%.	42/5 0.4kb 81.9%
Q4	297/356 107kb 84.0%	119/61 8kb 83.7%	44/22 1kb 82.4%	26/13 0.5kb 81.8%	14/5 0.1kb 78.3%

Table 2: Number of profiles, average profile length, size of profile codebook and word accuracy for different quantisation and profile compression settings.

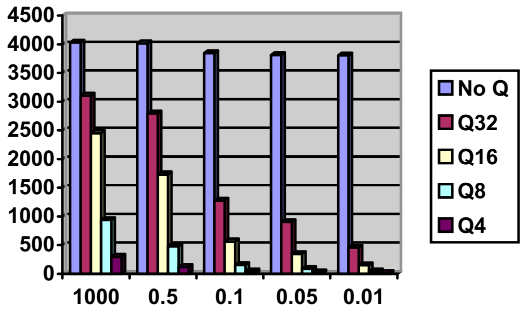

Figure 1: Number of unique profiles for different profile compression scales.

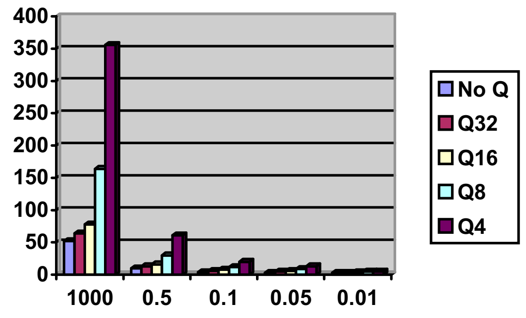

Figure 2: Average profile length for different compression scales.

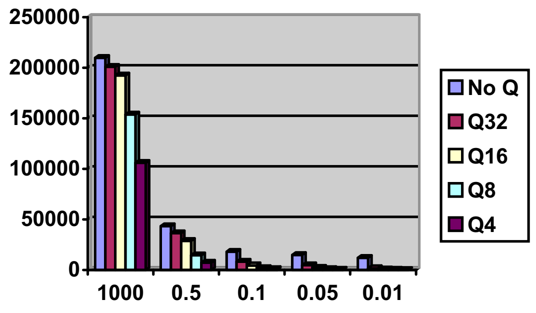

Figure 3: Size of profile codebook (sum of profile lengths) for different profile compression scales.

6. DISCUSSION AND CONCLUSIONS

In this paper a profile based structure was suggested for storing n-gram language models. The main purpose of the structure is to allow compression of the n-gram model by compressing and sharing profiles through a profile codebook. It is interesting to note, that for the language model used here, the profile format achieves a significant memory reduction even when compression or quantisation is not used. If all n-gram probabilities were unique, then the profile codebook would have 23k entries, and on average each profile would have a lengh of 11.9. This means that the profile codebook would require 1.1 Mb storage (see eq. 3) – but in fact it turns out that only 4033 profiles are unique (average length 52) and consequently only 436 kb storage is needed.

Probability quantisation is used to reduce the number of unique probabilities that have to be represented. In this study, quantisation was achieved using a k-means clustering algorithm on the probabilities in the codebook. Only bigram probabilities were quantised – unigram and backoff probabilities can be quantised also, but because they are numerically in a different range, it is better to use separate codebooks for this purpose.

Quantisation is very effective in reducing the memory requirements for storing probabilities in a language model. 8-bit quantisation (256 elements) is easily achieved, and even 5 and 4-bit codebooks (32 and 16 elements) produce little degradation in word accuracy. With 5-bit indexes, the profile codebook is reduced in size by a factor of two while word accuracy is only marginally lower (85.1%->85.0%).

Profile compression used on its own is more effective in reducing the memory footprint needed for representing the profile codebook than is quantisation used on its own – roughly by a factor of two: at a reduction to 25% of the original codebook size, the word accuracy is practically the same (0.1% lower absolute).

Quantisation and profile compression are not required to be used independently – the profile encoding of the n-gram probabilities benefits directly from quantisation in the sense that far fewer unique profiles are needed when quantisation is used – e.g. for 5 and 4-bit VQ codebooks and loss less profile encoding, the number of profiles is reduced by respectively 30% and 40%. With lossy profile compression the memory needed for storing probabilities can be reduced by a factor 12 (436kb/36kb) with only 0.1% absolute loss of word accuracy (Q16, S=0.5). For the most heavily compressed language model used here (Q4, S=0.01), the reduction is of a factor 3000 (from 436kb to 135byte) with a 6.8% (85.1%->78.3%) absolute loss in accuracy. In practice it is probably not worthwhile to compress this heavily, because long before this point, the size of the profile codebook has become insignificant in comparison to the "word-pair" section of the LM (603kb).

10. ACKNOWLEDGEMENTS

This work has partially been funded by the European Union under the integrated project TC-STAR - Technology and Corpora for Speech-to-Speech Translation - (IST-2002-FP6-506738, http://www.tc-star.org).

11. REFERENCES

[1] R. Rosenfeld, "Two Decades of Statistical Language Modelling: Where Do We Go from Here?", Proc. of the IEEE Vol. 88(8), pp. 1270-1278, 2000

[2] S.M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. 35(3), pp. 400-401, 1987

[3] F. Jelinek, "Self Organized Language Modeling for Speech Recognition", in Readings in Speech Recognition, Alex Waibel and Kai-Fu Lee (Eds.), Morgan Kaufmann, 1989

[4] K. Seymore and Ronald Rosenfeld, "Scalable Backoff Language Models", Proc. of ICSLP, Vol. 1, pp. 232-235, 1996

[5] A. Stolcke, "Entropy-based Pruning of Backoff Language Models", Proc. of DARPA Broadcast News Transcription and Understanding Workshop, pp. 270-274, 1998

[6] E. Whittaker, B. Raj, "Quantization-based Language Model Compression", Proc. Eurospeech, pp. 33-36, 2001

[7] P. Witschel et al., "POS-based Language Models for Lange Vocabulary Speech Recognition on Embedded Systems", Proc. of Interspeech2005, pp. 1333-1336

[8] P.F. Brown, V.J. DellaPietra, P.V. deSouza, J.C. Lai, R.L Mercer, "Class-based n-gram Models of Natural Language", Computational Linguistics, Vol 18, pp. 467-479, 1990

[9] R. Kneser and H. Ney, "Improved Clustering Techniques for Class-Based Statistical Language Modelling", Proceedings of the European Conference on Speech Communication and Technology, pp. 973-976, 1993.

[10] A. Gruenstein, C. Wang and S. Seneff, "Context-Sensitive Statistical Language Modeling", Proceedings of Interspeech 2005 - Eurospeech, pp. 17-20, 2005.