Compact n-gram models by incremental growing and clustering of histories

Sami Virpioja and Mikko Kurimo

Adaptive Informatics Research Centre
Helsinki University of Technology, Finland
sami.virpioja@tkk.fi, mikko.kurimo@tkk.fi

Abstract

This work concerns building n-gram language models that are suitable for large vocabulary speech recognition in devices that have a restricted amount of memory and space available. Our target language is Finnish, and in order to evade the problems of its rich morphology, we use sub-word units, morphs, as model units instead of the words. In the proposed model we apply incremental growing and clustering of the morph n-gram histories. By selecting the histories using *maximum a posteriori* estimation, and clustering them with information radius measure, we obtain a clustered varigram model. We show that for restricted model sizes this model gives better cross-entropy and speech recognition results than the conventional n-gram models, and also better recognition results than non-clustered varigram models built with another recently introduced method.

Index Terms: language models, clustering, information radius, speech recognition

1. Introduction

Statistical language models try to determine the probability distribution $P(S)$ over text strings S, which are usually observed as sequences of words, $w_1 \dots w_n = w_1^n$. They are a vital part of e.g. speech recognition systems. A large number of statistical language models have been developed and studied, but the most widely used models are still the n-gram models, based on the assumption that the probability of a word depends only on the $n-1$ previous words. When an n-gram model is trained, all different n-grams up to length n are collected from the selected training corpus, and probability for each n-gram is estimated by the ratio of occurrences of the n-gram w_{i-n+1}^{i} and its history w_{i-n+1}^{i-1}. Estimates are then smoothed, giving some probability mass to unseen events, which are then estimated by the distribution of shorter n-grams. However, the number of different n-grams in a large corpus can be very high, and also the sizes of the n-gram models easily grow impractically large. The problem is that on-line speech recognition requires models that it fit well into the work memory of the used device. We can well assume that the variety of devices will grow to include also personal digital assistants and mobile phones. Thus the resources will be limited also in the future, regardless of their overall growth.

Trivial ways of building smaller n-gram models are to restrict either the training data or the maximum n-gram length. However, it is clear that neither of those will get us a good model. Pruning, i.e. removing some of the n-grams from a full n-gram model, is a better way of reducing the model size. One efficient pruning method is entropy-based pruning [1]. There relative entropy resulting from removing each single n-gram from the model is calculated, and those n-grams that increase the entropy least are pruned. There are also ways of getting a pruned model without first estimating the full model. E.g. Siivola and Pellom [2] show a way of building the model incrementally from a unigram model by adding suitable sets of n-grams at a time. The result is so called *varigram* model, where there is no preset limit for the lengths of the n-grams included in the model. Thus the longer contexts are used only there where they are really needed.

Another common approach to the data sparseness problem is clustering. Traditionally, clustering has been added into n-gram models by the means of clustering the model units, i.e. words. The basic class-based n-gram model was presented in [3], often referred as IBM model. Some extensions for the model has been reported to work significantly better, see e.g. [4]. In addition to the benefit of saving space and memory, one can hope that the decrease in the number of parameters helps to avoid overlearning.

Even if pruning and clustering are in use, applying standard n-gram models to languages that are highly-inflecting or compounding has been very hard, as the number of different word forms in a large corpus may be enormous. That results in such sparseness that word-based models are inconvenient. One simple solution is to model smaller segments of text instead of words. A good choice for the segments is shown to be morpheme-like segments found statistically. E.g. for Finnish, utilization of morphs found by the Morfessor algorithm [5] has improved direct prediction performance compared to words, and reduced error rates in large vocabulary speech recognition compared to either words or grammatical morphs [6]. Statistical morphs have also been tested in Turkish and Estonian speech recognition with similar results [7].

For finding a desired balance between the model size and accuracy, we propose a method that makes use of the morph based language modeling, and combines both pruning and clustering. The main issue is applying clustering to a morph-based model. When we use morphs, the number of model units can be very limited. The benefit of clustering an already small number of units, at least into hard clusters, is not very promising. Instead it is more sensible to cluster some larger entities. A logical things to cluster in n-gram models are the histories of the n-grams, i.e. the sequences that are used predict the next model unit. For each history cluster we can then estimate a collective prediction distribution. The histories that we cluster are selected by the means of incremental building of the model.

1.1. Related work

Several approaches for building varigram models have been presented. We already mentioned a recent one [2]. Naturally, also various methods for clustering the model units have been developed. Some, such as Niesler and Woodland [8], even combine

varigram models and clustering. However, there is not much research done on models where n-gram histories would be clustered into equivalence classes in a similar manner that we will do. Goodman [9] mentions the idea in the section of his survey that concerns clustering, but solely states that there are many difficult issues to solve in it.

The closest related method that we are aware of is presented by Siu and Ostendorf [10]. They used clustering of n-gram histories in order to handle conversational speech characteristics such as filler words and repetitions. The search algorithm compares each history only to ones that are a small variation of its context, which is more limited than our approach. They reported that they could use a considerable smaller models in order to get the same perplexity and speech recognition results as for standard n-gram models. However, most of the benefit was due to using a varigram model instead of a full n-gram model.

Another exception is the work by Xu and Jelinek [11], where randomly grown Decision Trees were used to cluster word histories. Their language model outperformed traditional n-gram model in both perplexity evaluation and speech recognition.

2. An n-gram model of clustered histories

Let us denote the units of the model as w_i and each included history as h_j, $j \in \{1, \ldots, N\}$. We try to find a set of clusters of histories c_k, $k \in \{1, \ldots, C\}$, so that each history belongs to one cluster. In addition, we assume that the next unit depends only on the cluster of the history, not on the history itself. When predicting the next unit w for the known history h, we get

$$P(w \mid h) = P(w \mid c(h)), \tag{1}$$

where $c(h)$ denotes the cluster of the history h.

For this kind of model to work in practice, those histories that belong to the same cluster should have as similar prediction distributions as possible. There are several possible measures to compare the similarity of the distributions. We use *information radius*, which is a bounded metric, and reported to work well on auto-induction of semantic classes [12]. Information radius between distributions p and q is

$$\mathrm{IRad}(p \,||\, q) = D(p \,||\, \frac{p+q}{2}) + D(q \,||\, \frac{p+q}{2}), \tag{2}$$

where $D(x \,||\, y)$ is the Kullback-Leibler divergence of the distributions.

2.1. MAP estimation of the model

In a general level, conventional n-gram models are based on maximum likelihood (ML) estimates. I.e., they try to find a model G_{ML} that maximizes the likelihood of the observed data O:

$$G_{\mathrm{ML}} = \arg\max_G P(O \mid G) \tag{3}$$

The $P(O \mid G)$ is just the likelihood of the training data according to the model:

$$P(O \mid G) = \prod_i P(w_i \mid h_i, G), \tag{4}$$

where (w_i, h_i) are the n-grams in O. When we maximize the likelihood $P(O \mid G)$ we minimize the coding length L of the data given the model, as known from the information theory. But this should not be what we really want: If the model is flexible enough, it will overlearn the training data, and not generalize beyond it.

Instead of $P(O \mid G)$, we should be more interested in finding the model G_{MAP} that is the most probable when we know the data. Applying Bayes' theorem, we get

$$\begin{aligned} G_{\mathrm{MAP}} &= \arg\max_G P(G \mid O) \\ &= \arg\max_G \frac{P(G)P(O \mid G)}{P(O)}. \end{aligned} \tag{5}$$

This estimate is called *maximum a posteriori* (MAP) estimate. Note that the probability of the data $P(O)$ is not affected by G.

With a suitable prior probability $P(G)$, MAP estimation has a direct connection to the Minimum Description Length (MDL) principle, which is used e.g. in [2] to decide which n-grams are added to the language model. An introduction to the connection between Bayesian and MDL frameworks is found e.g. in [13].

2.2. Search algorithm

The components of our model were the n-gram histories, and C clusters c_k together with emission distributions $P(w_i \mid c_k)$. Each history has a link to the cluster to which it belongs. The basic algorithm for constructing the model G is the following:

1. Set $n = 1$
2. For each n-gram history w_{i-n+1}^{i-1} (or empty history if $n = 1$) in the training data O:
   a. If $n > 2$ and neither of the $(n-1)$-gram histories w_{i-n+2}^{i-1} or w_{i-n+1}^{i-2} are in the model, skip the history.
   b. Select the cluster c_k in the model that minimizes the information radius between ML estimates of $P(w_i \mid c_k)$ and $P(w_i \mid w_{i-n+1}^{i-1})$.
   c. Calculate $\Delta \log P(G \mid O) = \Delta \log P(G) + \Delta \log P(O \mid G)$ for the cases where (1) w_{i-n+1}^{i-1} is added to the model into the cluster c_k, and (2) a new cluster containing w_{i-n+1}^{i-1} is added to the model.
   d. Do (1) or (2) depending on which increases the posterior probability $P(G \mid O)$ more. If neither does, skip the history.
3. If new histories were added into the model, increase n by one and go to 2. Otherwise stop.

We use information radius to select the best matching cluster instead of calculating the change in posterior probability mostly because it is computationally more efficient. The search can be optimized more by first testing that the most probable units that follow the cluster and the new history are similar.

The idea behind the step 2a. is to speed up the training even more. Calculating prediction distributions and searching for nearest clusters cannot be done to every n-gram of the training data, at least if we want n to be large. Instead we grow those n-grams that are already accepted in the model. Thus we assume that if some history was considered useful in predictions, longer histories that include it may also be useful.

2.3. Model priors

Next we need the prior probabilities of the model, $P(G)$. We assume a predefined lexicon of model units. In the model we save N histories from the training data and the units of which they consist

of. Then we have C clusters, and save the cluster of each history. Moreover, each cluster has a prediction distribution for a number of units. Each saved parameter has a prior distribution (or a coding scheme, if we use the MDL framework). Prior probability of the model is the product of all priors of the parameters.

Our goals in the selection of the priors were very simple: First, give smaller probabilities to models that have larger number of histories. Second, give smaller probabilities for clusters whose distributions contain a wide range of morphs (and thus consume much memory). Otherwise, give equal probabilities to all possibilities.

Thus the more histories and clusters there will be in the model, the smaller $P(G)$ will be. On the other hand, $P(O \mid G)$ will be larger, so a compromise between the size and the accuracy of the model is made. Since we are not actually interested in finding the optimal coding for the training data, we can also weight the two parts by selecting some constant α and setting $P(G \mid O)$ proportional to $P(G)^{\alpha} P(O \mid G)$. This serves as a practical way to get larger or smaller models without redesigning the prior.

3. Experiments

In our experiments we compared three kind of language models: Baseline n-gram models built with the SRI toolkit [14], varigram models built with the growing algorithm [2], and the proposed clustered varigram models. First we calculated cross-entropy for held-out data sets, and then used the models in a speech recognition system.

Our main Finnish text corpora are from the language bank of Finnish IT center for science (CSC)[1]. For the training material we selected several books and magazines, total 8 600 000 words. A morph lexicon of 2 113 units was estimated from it using the Morfessor software [5]. We also trained some growing varigram models with the full 150 million word data for comparison, still using the same set of morphs.

Cross-entropy calculates the average number of bits needed to encode one data unit (morph) using the model, and is logarithm of the more commonly used measure, perplexity[2]. However, decrease in cross-entropy may generally reflect more accurately the potential decrease in error rate in speech recognition [9]. For cross-entropy test we had two separate corpora: First consisted of articles of one year of the tabloid magazine Iltalehti (100 000 words). Second was a book (50 000 words) that was used also in speech recognition tests.

For speech recognition experiments we used the speech recognition system of the Adaptive Informatics Research Centre (AIRC) [7]. The book in audio form contained 13h of speech read by one female speaker [6]. Acoustic models were trained for the speaker using the first 11 hours. Speech recognition results were measured by word error rate, i.e. the precentage of words not recognized properly from the test set.

3.1. Cross-entropy results

For cross-entropy evaluation we built up baseline 1-gram, 2-gram and 3-gram models, and several growing varigram and clustered varigram models of different sizes. Our area of interest in the language model sizes was from 200 000 to one million parameters. For the both varigram types, the size could be varied by adjusting the weight parameter, whereas size of the baseline models could not be adjusted more than by selecting the n-gram length.

[1] `http://www.csc.fi/kielipankki/`

[2] The reported values can be (approximately) converted to word-based perplexities with the formula $2^{3.45H}$, where H is the morph-based entropy. The coefficient 3.45 is the average number of morphs per word.

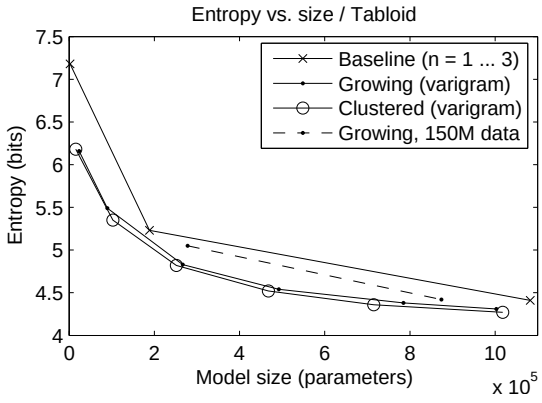

Figure 1: *Cross-entropies of the different models on the tabloid data. Measurement points of the growing and clustered varigram model correspond to different parameter values. For the baseline n-gram they are 1-, 2-, and 3-gram models. Cross-entropy of 4 bits corresponds to normalized perplexity of about 14 000, and 5 bits to 156 000.*

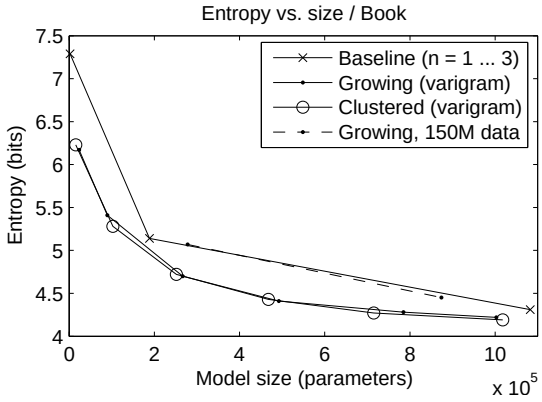

Figure 2: *Cross-entropies of the different models on the book data.*

The cross-entropy results for the two tasks are plotted in Figures 1 and 2. Both varigram model types clearly win the equal sized baseline models. We also see that the clustered model is somewhat better than the non-clustered varigram model with larger models and in the harder task (tabloid magazine). In addition, using the full 150 million word training data did not help with less than million parameter models. If also the model size was allowed to grow almost 20 times larger, we reached cross-entropy 3.56 for the tabloid data and 3.81 for the book data.

3.2. Speech recognition results

For speech recognition tests we selected baseline 2-gram and 3-gram models, and three models for each varigram model types. The smallest varigram models had about 250 000 parameters, middle sized models had about 500 000 parameters and the largest models had about one million parameters.

For each individual model, recognition parameters were first optimized using the development set. For the test set we varied

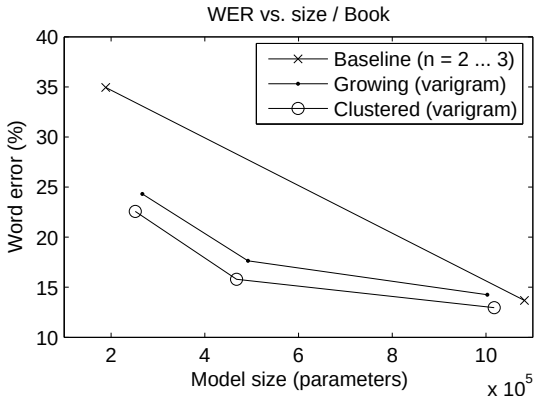

Figure 3: *The best obtained word error rates of the speech recognition experiments for models of different type and size.*

beam pruning settings and allowed real-time factors up to 8 for obtaining the best recognition results. However, factors smaller than two were enough for almost as good rates.

The best word error rates are plotted in in Figure 3. Resemblance to cross-entropy figures is clear. However, we see that the clustered varigram model is now clearly better than the growing varigram. For half million parameter models, growing model achieves error rate 17.64% and clustered model 15.80%. On the other hand, baseline 3-gram model gets nearer to both varigram models than it managed in cross-entropies.

4. Conclusions and discussion

We have proposed a way of constructing more compact n-gram models by clustering of n-gram histories. Combined with reasonable smoothing, pruning and selection of model units, we managed to clearly improve both cross-entropy and speech recognition results for small models compared to the baseline n-gram models. In addition, the model outperformed also the growing varigram model, which included all other discussed improvements except the clustering.

The clustered model worked relatively better in speech recognition than in direct text prediction. A non-clustered model gives naturally more accurate predictions, as clustering approximates the probabilities by using the same parameters for several distributions. However, such accuracy is rarely needed in applications such as speech recognition, where the probabilities are combined with some external information. An important side effect of the clustering is that the model can generalize the distributions, instead of relying only on the exact examples in the training data.

Problem of the optimal solution for clustering the histories is exponential: N histories can be divided in C clusters in $O(C^N)$ ways, which is too much even for solution with a couple of clusters. Every realistic algorithm, such as ours, will only find a local optimum. The same applies to the selection of the histories to include in the model.

One clear, and presumably even critical, improvement to the proposed model would be to use soft clustering instead of the hard one. That would allow smaller number of clusters but still a more flexible model, as there would be no need for exactly similar distributions among the histories. Optimizing the soft clustering is clearly even a more challenging task, but we believe that the useful methods already exist, e.g. in the field of blind source separation. How to apply the methods so that they cope with the high dimensionality of the language data is the main issue in our future research.

5. Acknowledgements

This work was funded by Nokia Oy under the project *Search for Personal Media Content* (SEPEMCO). We thank Mathias Creutz and Vesa Siivola for their useful comments and help.

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 and B. L. Pellom, "Growing an n-gram language model," in *Proc. Interspeech*, 2005.

[3] P. F. Brown, V. J. DellaPietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[4] J. Gao, J. T. Goodman, G. Cao, and H. Li, "Exploring asymmetric clustering for statistical language modeling," in *Proc. ACL*, 2002, pp. 183–190.

[5] M. Creutz and K. Lagus, "Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor," Tech. Rep. A81, Publications in Computer and Information Science, Helsinki University of Technology, 2005.

[6] T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen, "Unlimited vocabulary speech recognition with morph language models applied to Finnish," *Computer Speech and Language*, 2005, In press, Available online.

[7] M. Kurimo, A. Puurula, E. Arisoy, V. Siivola, T. Hirsimäki, J. Pylkkönen, T. Alumäe, and M. Saraclar, "Unlimited vocabulary speech recognition for agglutinative languages," in *Proc. HLT-NAACL*, 2006.

[8] T. R. Niesler and P. C. Woodland, "A variable-length category-based n-gram language model," in *Proc. ICASSP*, 1996.

[9] J. T. Goodman, "A bit of progress in language modeling — extended version," Tech. Rep. MSR-TR-2001-72, Microsoft Research, 2001.

[10] M. Siu and M. Ostendorf, "Variable n-grams and extensions for conversational speech language modeling," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 8, no. 1, pp. 63–75, 2000.

[11] P. Xu and F. Jelinek, "Random forests in language modeling," in *Proc. EMNLP*, 2004.

[12] A. Pargellis, E. Fosler-Lussier, A. Potamianos, and C.-H. Lee, "A comparison of four metrics for auto-inducing semantic classes," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, 2001.

[13] S. F. Chen, *Building Probabilistic Models for Natural Language*, Ph.D. thesis, Harvard University, 1996.

[14] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proc. ICSLP*, 2002, pp. 901–904.