On Growing and Pruning Kneser–Ney Smoothed N-Gram Models

Vesa Siivola, Teemu Hirsimäki, and Sami Virpioja

***Abstract*—N-gram models are the most widely used language models in large vocabulary continuous speech recognition. Since the size of the model grows rapidly with respect to the model order and available training data, many methods have been proposed for pruning the least relevant N-grams from the model. However, correct smoothing of the N-gram probability distributions is important and performance may degrade significantly if pruning conflicts with smoothing. In this paper, we show that some of the commonly used pruning methods do not take into account how removing an N-gram should modify the backoff distributions in the state-of-the-art Kneser–Ney smoothing. To solve this problem, we present two new algorithms: one for pruning Kneser–Ney smoothed models, and one for growing them incrementally. Experiments on Finnish and English text corpora show that the proposed pruning algorithm provides considerable improvements over previous pruning algorithms on Kneser–Ney-smoothed models and is also better than the baseline entropy pruned Good–Turing smoothed models. The models created by the growing algorithm provide a good starting point for our pruning algorithm, leading to further improvements. The improvements in the Finnish speech recognition over the other Kneser–Ney smoothed models are statistically significant, as well.**

***Index Terms*—Modeling, natural languages, smoothing methods, speech recognition.**

I. Introduction

N-GRAM models are the most widely used language models in speech recognition. Since the size of the model grows fast with respect to the model order and available training data, it is common to restrict the number of N-grams that are given explicit probability estimates in the model. A common approach is to estimate a full model containing all N-grams of the training data up to a given order and then remove N-grams according to some principle. Various methods such as count cutoffs, *weighted difference pruning* (WDP) [1], *Kneser pruning* (KP) [2], and *entropy-based pruning* (EP) [3] have been used in the literature. Experiments have shown that more than half of the N-grams can be removed before the speech recognition accuracy starts to degrade.

Another important aspect in N-gram language modeling is smoothing to avoid zero probability estimates for unseen data. Numerous smoothing methods have been proposed in the past, but the extensive studies by Chen and Goodman [4], [5] showed that a variation of Kneser–Ney smoothing [6] outperforms other smoothing methods consistently.

In this paper, we study the interaction between pruning and smoothing. To our knowledge, this interaction has not been studied earlier, even though smoothing and pruning are widely used. We demonstrate that EP has some assumptions that conflict with the properties of Kneser–Ney smoothing, but work well for the Good–Turing smoothed models. KP, on the other hand, takes better into account the underlying smoothing, but has other approximations in the pruning criterion. We then describe two new algorithms for selecting N-grams of Kneser–Ney smoothed models more efficiently. The first algorithm prunes individual N-grams from models, and the second grows models incrementally starting from a 1-gram model. We show that the proposed algorithms produce better models than the other pruning methods.

The rest of the paper is organized as follows. Section II surveys earlier methods for pruning and growing N-gram models, and other methods for modifying the context lengths of N-gram models. Similarities and differences between the previous work and the current work are highlighted. Section III describes the algorithms used in the experiments, and Section IV presents the experimental evaluation with discussion.

II. Comparison to Previous Work

A. Methods for Pruning Models

The simplest way for reducing the size of an N-gram model is to use count cutoffs: An N-gram is removed from the model if it occurs fewer than T times in the training data, where T is a fixed cutoff value. Events seen only once or twice can usually be discarded without significantly degrading the model. However, severe pruning with cutoffs typically gives worse results than other pruning methods [7].

WDP was presented by Seymore and Rosenfeld [1]. For each N-gram in the model, WDP computes the log probability given by the original model and a model from which the N-gram has been removed. The difference is weighted by a Good–Turing discounted N-gram count, and the N-gram is removed if the weighted difference is smaller than a fixed threshold value. In their experiments (presumably with Good–Turing smoothed models), the weighted difference method gave better results than count cutoffs.

Kneser [2] proposes a similar method for pruning N-gram models. The pruning criterion used in KP also computes the weighted difference in log probability when an N-gram is pruned. The difference is computed using an absolute discounted model and weighted by the probability given by the

Manuscript received November 3, 2006; revised January 27, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Bill Byrne.

The authors are with the Adaptive Informatics Research Centre, Helsinki University of Technology, FI-02015 HUT, Finland (Vesa.Siivola@tkk.fi; Teemu.Hirsimaki@tkk.fi; Sami.Virpioja@tkk.fi).

Digital Object Identifier 10.1109/TASL.2007.896666

model. Kneser also shows that using modified backoff distributions along the lines of the original Kneser–Ney smoothing improves the results further.

EP presented by Stolcke [3] is also closely related to WDP. While WDP (and KP) only takes into account the change in the probability of the pruned N-gram, EP also computes how the probabilities of other N-grams change. Furthermore, instead of using the discounted N-gram count for weighting the log probability difference, EP uses the original model for computing the probability of the N-gram. Hence, EP can be applied to a ready-made model without access to the count statistics. In Stolcke's experiments with Good–Turing smoothed models, EP gave slightly better results than WDP.

In this paper, we propose a method called *revised Kneser pruning* (RKP) for pruning Kneser–Ney smoothed models. The method takes the properties of Kneser–Ney smoothing into account already when selecting the N-grams to be pruned. The other methods either ignore the smoothing method when selecting the N-gram to be pruned (KP) or ignore the fact that as an N-gram gets pruned, the lower-order probability estimates should be changed (WDP, EP). We use the original KP and EP as baseline methods, and they are described in more detail in Section III.

B. Methods for Growing Models

All the algorithms mentioned in the previous section assume that the N-gram counts are computed from the training data for every N-gram up to the given context length. Since this becomes computationally impractical if long contexts are desired, various algorithms have been presented for selecting the N-grams of the model incrementally, thus avoiding computing the counts for all N-grams present in the training data.

Ristad and Thomas [8] describe an algorithm for growing N-gram models. They use a greedy search for finding the individual candidate N-grams to be added to the model. The selection criterion is a minimum description length (MDL)-based cost function. Ristad and Thomas train their letter N-gram model using 900 000 words. They get significant improvements over their baseline N-gram model, but it seems their baseline model is not very good, as its performance actually gets significantly worse when longer contexts are used.

Siu and Ostendorf [9] present their N-gram language model as a tree structure and show how to combine the tree nodes in several different ways. Each node of the tree represents an N-gram context and the conditional N-gram distribution for the context. Their experiments show that the most gain can be achieved by choosing an appropriate context length separately for each word distribution. They grow the tree one distribution at a time, and contrary to the other algorithms mentioned here, contexts are grown toward the past by adding new words to the beginning of the context. Their experiments on a small training data (fewer than three million words) show that the model's size can be halved with no practical loss in performance.

Niesler and Woodland [10] present a method for backing off from standard N-gram models to cluster models. Their paper also shows a way to grow a class N-gram model which estimates the probability of a cluster given the possible word clusters of the context. The greedy search for finding the candidates to be added to the model is similar to the one by Ristad and Thomas. Whereas Ristad and Thomas add individual N-grams, Niesler and Woodland add conditional word distributions for N-gram contexts, and then prune away unnecessary N-grams.

To our knowledge, no methods for growing Kneser–Ney smoothed models have been proposed earlier. In this paper, we present a method for estimating variable-length N-gram models incrementally while maintaining some aspects of Kneser–Ney smoothing. We refer to the algorithm as *Kneser–Ney growing* (KNG). It is similar to the growing method presented earlier [11], except that RKP is used in the pruning phase. Additionally, some mistakes in the implementation have been corrected. The original results were reasonably good, but the correct version gives clearly better results. The growing algorithm is similar to the one by Niesler and Woodland. They use the leaving-one-out cross validation for selecting the N-grams for the model, whereas our method uses a MDL-based cost criterion. The MDL criterion is defined in a simpler manner than in the algorithm by Ristad and Thomas, where a tighter and more theoretical criterion was developed. We have chosen a cost function that reflects how N-gram models are typically stored in speech recognition systems.

C. Other Related Work

Another way of expanding context length of the N-gram models is to join several words (or letters) to one token in the language model. This idea is presented for example in a paper on word clustering by Yamamoto *et al.* [12]. Deligne and Bimbot [13] study how to combine several observations into one underlying token. The opposite idea, splitting words into subword units to improve the language model, has also been studied. In our Finnish experiments, we use the algorithm presented by Creutz and Lagus [14] for splitting words into morpheme-like units.

Goodman and Gao [7] show that combining clustering and EP can give better results than pruning alone. In the current work, however, we only consider models without any clustering.

Virpioja and Kurimo [15] describe how variable-length N-gram contexts consisting of subword units can be clustered to achieve some improvements in speech recognition. They have also compared the performance to the old version of KNG with a relatively small data set of around ten million words, and show that the clustering gives better results with the same number of parameters. Recent preliminary experiments suggest that if RKP is applied also to the clustered model, the improvement in perplexity is about as good as it was for the nonclustered algorithm.

Bonafonte and Mario [16] present a pruning algorithm, where the distribution of a lower order context is used instead of the original if the pruning criterion is satisfied. For their pruning criterion, they combine two requirements: The frequency of the context must be low enough (akin to count cutoffs) or the Kullback–Leibler divergence between the distributions must be small enough. The combination of these two criteria is shown to work better than either of the criteria alone when the models were trained with a very small training set (14 000 sentences, 1300 words in the lexicon).

III. Algorithms

A. Interpolated Kneser–Ney Smoothing

Let w be a word and $\boldsymbol{h}$ the history of words preceding w. By $\hat{\boldsymbol{h}}$ we denote the history obtained by removing the first word in the history $\boldsymbol{h}$. For example, with the three-word history $\boldsymbol{h} = abc$ and word $w = d$, we have N-grams $\boldsymbol{h}w = abcd$ and $\hat{\boldsymbol{h}}w = bcd$. The number of words in the N-gram $\boldsymbol{h}w$ is denoted by $|\boldsymbol{h}w|$. Let $C(\boldsymbol{h}w)$ be the number of times $\boldsymbol{h}w$ occurs in the training data.

Interpolated Kneser–Ney smoothing [4] defines probabilities $P_{\mathrm{KN}}(w|\boldsymbol{h})$ for an N-gram model of order N as follows:

$$P_{\mathrm{KN}}(w|\boldsymbol{h}) = \frac{\max\left\{0, C'(\boldsymbol{h}w) - D_{|\boldsymbol{h}|}\right\}}{S(\boldsymbol{h})} + \gamma(\boldsymbol{h}) P_{\mathrm{KN}}(w|\hat{\boldsymbol{h}}). \tag{1}$$

The modified counts $C'(\boldsymbol{h}w)$, the normalization sums $S(\boldsymbol{h})$, and the interpolation weights $\gamma(\boldsymbol{h})$ are defined as

$$C'(\boldsymbol{h}w) = \begin{cases} 0, & \text{if } |\boldsymbol{h}w| > N \\ C(\boldsymbol{h}w), & \text{if } |\boldsymbol{h}w| = N \\ \left|\{v : C(v\boldsymbol{h}w) > 0\}\right|, & \text{otherwise} \end{cases} \tag{2}$$

$$S(\boldsymbol{h}) = \sum_v C'(\boldsymbol{h}v) \tag{3}$$

$$\gamma(\boldsymbol{h}) = \frac{\left|\{v : C'(\boldsymbol{h}v) > 0\}\right| D_{|\boldsymbol{h}|}}{S(\boldsymbol{h})}. \tag{4}$$

Order-specific discount parameters D_i can be estimated on held-out data. In (2), $C(\boldsymbol{h}w)$ also has to be used for N-grams $\boldsymbol{h}w$ that begin with the sentence start symbol because no word can precede them.

The original intention of Kneser–Ney smoothing is to keep the following marginal constraints (see [6] for the original backoff formulation, and [5] for the interpolated formulation)

$$\sum_v P(v\boldsymbol{h}w) = P(\boldsymbol{h}w). \tag{5}$$

Despite the intention, the smoothing satisfies the above constraints only approximately. In order to keep the marginals exactly, maximum entropy modeling can be used (see [17], for example), but the computational burden of maximum entropy modeling is high.

For clarity, the above equations show Kneser–Ney smoothing with only one discount parameter for each N-gram order. James [18] showed that the choice of discount coefficients in Kneser–Ney smoothing can affect the performance of the smoothing. In the experiments we used modified Kneser–Ney smoothing [4] with three discount parameters for each N-gram order: one for N-grams seen only once, one for N-grams seen only twice, and one for N-grams seen more than two times. We use numerical search to to find discount parameters that maximize the probability of the held-out data.

B. Entropy-Based Pruning

Stolcke [3] described EP for backoff language models. For each N-gram $\boldsymbol{h}w$ in model M, the pruning cost $d(\boldsymbol{h}w)$ is computed as follows:

$$d(\boldsymbol{h}w) = \sum_v P_M(\boldsymbol{h}v) \log \frac{P_M(v|\boldsymbol{h})}{P_{M'}(v|\boldsymbol{h})}. \tag{6}$$

P_M is the original model, and $P_{M'}$ corresponds to a model from which the N-gram $\boldsymbol{h}w$ has been removed (and backoff weight $\gamma(\boldsymbol{h})$ updated accordingly). The cost is computed for all N-grams, and then the N-grams which cost less than a fixed threshold are removed from the model. It was shown that the cost can be computed efficiently for all N-grams. Another strength of EP is that it can be applied to the model without knowing the original N-gram counts.

However, only Good–Turing smoothed models were used in the original experiments. In the case of Kneser–Ney smoothing, the lower-order distributions $P_{\mathrm{KN}}(w|\hat{\boldsymbol{h}})$ are generally not good estimates for the true probability $P(w|\hat{\boldsymbol{h}})$. This is because the lower-order distributions are in a way optimized for modeling probabilities of unseen N-grams that are not covered by the higher order of the model.[1] This property conflicts with EP in two ways. First, the selection criterion of EP weights the change in $P_M(c|ab)$ with the probability

$$P(abc) \approx P_M(a) P_M(b|a) P_M(c|ab) \tag{7}$$

which is not a good approximation with Kneser–Ney smoothing as discussed above. For the same reason, pruning $P_{\mathrm{KN}}(c|ab)$ may be difficult if $P_{\mathrm{KN}}(c|b)$ is not a good estimate for the true $P(c|b)$. Indeed, we will see in Section IV that an entropy-pruned Kneser–Ney model becomes considerably worse than an entropy-pruned Good–Turing model when the amount of pruning is increased.

C. Kneser Pruning

Kneser [2] also describes a general pruning method for backoff models. For an N-gram $\boldsymbol{h}w$, which is not a prefix of any $(n+1)$-g included in the model ($\boldsymbol{h}w$ is a *leaf N-gram*), the cost of pruning from the full model M is defined as

$$d_1(\boldsymbol{h}w) = P_M(\boldsymbol{h}w) \log \frac{P_M(w|\boldsymbol{h})}{\gamma_M(\boldsymbol{h}) P_M(w|\hat{\boldsymbol{h}})}. \tag{8}$$

The cost $d_2(\boldsymbol{h}w)$ for a *non-leaf N-gram*, is obtained by averaging $d_1(\boldsymbol{g})$ for N-grams $\boldsymbol{g}$ that have $\boldsymbol{h}w$ as prefix (including $\boldsymbol{h}w$).

Kneser also gives a formula for computing modified backoff distributions that approximate the same marginal constraints as the original Kneser–Ney smoothing

$$\begin{aligned} &P_{\mathrm{KP}}(w|\boldsymbol{h}) \\ &\quad = \frac{\max\left\{0, \sum\limits_{v\boldsymbol{h}w \notin M} C(v\boldsymbol{h}w) + D_{|\boldsymbol{h}w|} \sum\limits_{v\boldsymbol{h}w \in M} 1 - D_{|\boldsymbol{h}|}\right\}}{\sum\limits_{w'} \left(\sum\limits_{v\boldsymbol{h}w' \notin M} C(v\boldsymbol{h}w') + D_{|\boldsymbol{h}w'|} \sum\limits_{v\boldsymbol{h}w' \in M} 1 \right)} \\ &\qquad + \gamma_{\mathrm{KP}}(\boldsymbol{h}) P_{\mathrm{KP}}(w|\hat{\boldsymbol{h}}). \end{aligned} \tag{9}$$

The interpolation coefficient γ_{KP} can be easily solved from the equation to account for the discounted and pruned probability

[1] For example, this can be verified by training a 3-g model using Good–Turing and Kneser–Ney smoothing, and then computing log probability of test data using the 1-g and 2-g estimates only. The truncation degrades the performance of the Kneser–Ney smoothed model dramatically when compared to the Good–Turing smoothed model.

mass. The above formulation corresponds to the original definition,[2] except that the original formulation was for backoff model, while ours is for interpolated model and the discount term $D_{|\boldsymbol{h}|}$ is explicitly shown. As with Kneser–Ney smoothing, the marginal constraints are not satisfied exactly.

The criterion for selecting N-grams to be pruned contains the following approximations: The selection is made before any model modification takes place, and the criterion utilizes the difference between the log probability of the N-gram and its backed-off estimate for the full absolute discounted model. Only d_2 is updated during pruning. In practice, however, both the backoff coefficient and the backoff distribution may be considerably different in the final pruned model with modified backoff distributions.

We have implemented an interpolated version of the algorithm, since it has been shown that interpolated models generally work better [4]. It is not explicitly clear how KP should be implemented with three discounts per model order, so we implemented the original unmodified version (one discount per order). In practice, the difference between modified and unmodified models with large training data should be very small [5].

We conducted some preliminary experiments with different approximations for selecting the N-grams, and it seemed that the criterion could be improved. These improvements are implemented in the algorithm presented in the next section.

D. Revised Kneser Pruning

Since the original KP and EP ignore the properties of Kneser–Ney smoothing when selecting N-grams to be pruned, we propose a new algorithm that takes the smoothing better into account. The main motivation is that removing an N-gram from a Kneser–Ney smoothed model should change the lower-order distributions. The algorithm tries to maintain the following property of Kneser–Ney smoothing: As shown in (2), a backoff distribution of a Kneser–Ney-smoothed model does not use actual word counts. Instead, the number of unique words appearing before the N-gram are counted. For the highest-order N-grams, the actual counts from the training data are used. We can view the highest-order N-gram counts in the same way as the lower-order counts if we pretend that all $(n+1)$-grams have been pruned, and each appearance of the highest-order N-gram is considered to have a unique preceding word in the training data.

This property is maintained in the algorithm shown in Fig. 1. PRUNEGRAM($\boldsymbol{h}w$) describes how the counts $C'(\cdot)$ and normalization sums $S(\cdot)$ are modified when an N-gram $\boldsymbol{h}w$ is pruned. Before pruning, the first word of $\boldsymbol{h}w$ is considered as one unique preceding word for $\hat{\boldsymbol{h}}w$ in $C'(\hat{\boldsymbol{h}}w)$. After pruning $\boldsymbol{h}w$, all the $C'(\boldsymbol{h}w)$ instances of $\boldsymbol{h}w$ are considered having a new unique preceding word for $\hat{\boldsymbol{h}}w$. Thus, $C'(\hat{\boldsymbol{h}}w)$ is increased by $C'(\boldsymbol{h}w)-1$. Note that the condition on line 2 of PRUNEGRAM is always true if the model contains all N-grams from the training data. However, if model growing or count cutoffs are used, $C'(\hat{\boldsymbol{h}}w)$ may be zero even if $C'(\boldsymbol{h}w)$ is positive. Additionally, the sum

[2]In the original paper [2, Eq. 9], there are parentheses missing around $N(v, h_k, w) - d$ in the numerator and denominator.

```
PRUNEORDER(k, ε)
1  for {hw : |hw| = k ∧ C'(hw) > 0} do
2      logprob_0 ← C(hw) log_2 P_KN(w|h)
3      PRUNEGRAM(hw)
4      logprob_1 ← C(hw) log_2 P_KN(w|h)
5      if logprob_1 < logprob_0 − ε
6          undo previous PRUNEGRAM

PRUNEGRAM(hw)
1  L(h) ← L(h) + C'(hw)
2  if C'(ĥw) > 0
3     C'(ĥw) ← C'(ĥw) + C'(hw) − 1
4     S(ĥ) ← S(ĥ) + C'(hw) − 1
5  C'(hw) ← 0
```

Fig. 1. Pruning algorithm. Note that lines 3 and 6 in PRUNEORDER modify the counts $C'(\cdot)$, which also alters the estimate $P_{\mathrm{KN}}(w|\boldsymbol{h})$.

of pruned counts $L(\boldsymbol{h})$ is updated with $C'(\boldsymbol{h}w)$. The probabilities $P_{\mathrm{KN}}(w|\boldsymbol{h})$ are then computed as usual (1), except that the interpolation weight γ has to take into account the discounted and pruned probability mass:

$$\gamma(\boldsymbol{h}) = \frac{|\{v : C'(\boldsymbol{h}v) > 0\}|\, D_{|\boldsymbol{h}|} + L(\boldsymbol{h})}{S(\boldsymbol{h})}. \tag{10}$$

For each order k in the model, PRUNEORDER(k, ϵ) is called with a pruning threshold ϵ. Higher orders are processed before lower orders. For each N-gram $\boldsymbol{h}w$ at order k, we try pruning the N-gram (and modifying the model accordingly), and compute how much the log probability of the N-grams $\boldsymbol{h}w$ decreases in the training data. If the decrease is greater than the pruning threshold, the N-gram is restored into the model. Note that the algorithm also allows pruning non-leaf nodes of an N-gram model. It may not be theoretically justified, but preliminary experiments suggested that it can clearly improve the results. For efficiency, it is also possible to maintain a separate variable for $|\{v : C'(\boldsymbol{h}v) > 0\}|$ in the algorithm. After pruning, we re-estimate the discount parameters on a held-out text data. In contrast to EP, the counts are modified whenever an N-gram is pruned, so the pruning cannot be applied to a model without count information.

The pruning criterion used in PRUNEORDER has a few approximations. It only takes into account the change in the probability of the pruned N-gram. In reality, pruning N-gram $abcd$ alters $P_{\mathrm{KN}}(w|bc)$ directly for all w. The interpolation weights $\gamma(abc)$ and $\gamma(bc)$ are altered as well, so $P_{\mathrm{KN}}(w|\boldsymbol{h}bc)$ may change for all w and $\boldsymbol{h}$. For weighting the difference in log probability, we use the actual count C. This should be a better approximation for Kneser–Ney smoothed models than the one used by EP. The Good–Turing weighting, as used in WDP, would probably be better, but would make the model estimation slightly more complex, since the model is now originally Kneser–Ney smoothed.

Note that apart from the criterion for choosing the N-grams to be pruned, the proposed method is very close to KP. If we chose to prune the same set of N-grams, RKP would give almost the same probabilities as shown in (9); only the factor $D_{|\boldsymbol{h}w|}$ would be approximated as one. This approximation makes it easier to reoptimize the discount factors on a held-out text data after pruning. In our preliminary experiments, this approximation did not degrade the results.

```
GROWORDER(k, δ)
 1  for {h : |h| = k − 1 ∧ C′(h) > 0} do
 2      size_0 ← |{g : C′(g) > 0}|
 3      logprob_0 ← 0
 4      for w : C(hw) > 0 do
 5          logprob_0 ← logprob_0 + C(hw) log_2 P_KN(w|h)
 6      for w : C(hw) > 0 do
 7          ADDGRAM(hw)
 8      size_1 ← |{g : C′(g) > 0}|
 9      logprob_1 ← 0
10      for w : C(hw) > 0 do
11          logprob_1 ← logprob_1 + C(hw) log_2 P_KN(w|h)
12      logscost = size_1 log_2(size_1) − size_0 log_2(size_0)
13      sizecost ← (size_1 − size_0)α + logscost
14      if logprob_1 − logprob_0 − δ · sizecost ≤ 0
15          undo previous ADDGRAM(hw) for each w
16  re-estimate all discount parameters D_i

ADDGRAM(hw)
1  C′(hw) ← C(hw)
2  S(h) ← S(h) + C(hw)
3  if C′(ĥw) > 0
4      C′(ĥw) ← C′(ĥw) − C(hw) + 1
5      S(ĥ) ← S(ĥ) − C(hw) + 1
```

Fig. 2. Growing algorithm.

Thus, the main differences to KP are the following: We modify the model after each N-gram has been pruned, instead of first deciding which N-grams to prune and pruning the model afterwards. The pruning criterion uses these updated backoff coefficients and distributions. Lastly, the pruning criterion weights the difference in log probability by the N-gram count instead of the probability estimated by the model.

The method looks computationally slightly heavier than EP or WDP, since some extra model manipulation is needed. In practice, however, the computational cost is similar. The memory consumption and speed of the method can be slightly improved by replacing the weighting $C(hw)$ by $C'(hw)$ in line 2 and 4 of PRUNEORDER algorithm (Fig. 1), since then the original counts are not needed at all, and can be discarded. In our preliminary experiments, this did not degrade the results.

E. Kneser–Ney Growing

Instead of computing all N-gram counts up to certain order and then pruning, a variable-length model can be created incrementally so that only some of the N-grams found in the training data are taken into the model in the first place. We use a growing method that we call Kneser–Ney growing. KNG is motivated similarly to the RKP described in the previous section.

The growing algorithm is shown in Fig. 2. The initial model is an interpolated 1-g Kneser–Ney model. Higher orders are grown by GROWORDER(k, δ), which is called iteratively with increasing order $k > 1$ until the model stops growing. The algorithm processes each N-gram $\boldsymbol{h}$ already in the model at order k, and adds all $(k+1)$-grams $\boldsymbol{h}w$ present in the training data to the model, if they meet a cost criterion. The cost criterion is discussed below in more detail. The ADDGRAM$(\boldsymbol{h}w)$ algorithm shows how count statistics used in (1) are updated when an N-gram is added to the model.

Since the model is grown one distribution at time, it is still useful to prune the grown model to remove the individual unnecessary N-grams. Compared to pruning of full N-gram models, the main computational benefit of the growing algorithm is that counts $C(\boldsymbol{h}w)$ only need to be collected for histories $\boldsymbol{h}$ that are already in the model. Thus, much longer contexts can be brought into the model.

1) About the Cost Function for Growing: For deciding which N-grams should be added to the model, we use a cost function based on the MDL principle. The cost consists of two parts: the cost of encoding the training data ($logprob$), and the cost of encoding the N-gram model ($sizecost$). The relative weight of the model encoding is controlled by δ, which affects the size of the resulting model. The cost of encoding the training data is the log probability of the training data given by the current model. For the cost of encoding the model, we roughly assume the tree structure used by our speech recognition system (the structure is based on [19]). The cost of growing the model from N_{old} N-grams to N_{new} N-grams is then

$$\Delta\text{Cost} = \alpha(N_{\text{new}} - N_{\text{old}}) + N_{\text{new}} \log_2 N_{\text{new}} - N_{\text{old}} \log_2 N_{\text{old}} \tag{11}$$

where α is related to the number of bits required for storing each float with given precision. The first term assumes that constant amount of bits is required for storing the parameters of an N-gram, regardless of the N-gram order. The remaining terms take into account the tree structure for representing the N-gram indices (see [11] for details), but omitting them does not seem to affect the results. In practice, during the model estimation, the model is stored in a different structure where model manipulation is easy.

More compact representations can be formulated. Ristad and Thomas [8] show an elaborate cost function which they use for training letter-based N-gram models. Whittaker and Raj [19], [20], on the other hand, have used quantization and compression methods for storing N-grams compactly while maintaining reasonable access times.

In practice, however, pruning or growing algorithms are not used for finding the model with the optimal description length. Instead, they are used for finding a good balance between the modeling performance (or recognition accuracy) and memory consumption. Moreover, even if the desired model size was, say, only 100 MB, we probably want to create first as large model as we can (perhaps a few gigabytes with current systems), and then prune it to the desired size. The same applies for growing methods. It may be hard to grow an optimal model for 100 MB, unless one first creates a larger model to see which N-grams really should be omitted. In this sense, the main advantage of the growing algorithms may be the ability to create good initial models for pruning algorithms.

F. Some Words on the Computational Complexity

The limiting factors for the algorithms are either the consumed memory or the required processing power. All of the algorithms presented here can be implemented with similar data structures. For models containing equal amount of N-grams, the methods will end up using similar amounts of memory. When looking at the processor time, some algorithms are clearly simpler than the others. In practice though, they all scale similarly with the number of N-grams in the model. In our experiments,

the computation times of the methods were roughly equivalent using a computer with a 2-GHz consumer level processor and 10 GB of memory.

IV. Experiments

A. Setup and Data

The Finnish text corpus (150 million words) is a collection of books, magazines, and newspapers from the Kielipankki corpus [21]. Before training the language models, the words were split into subword units, which has been shown to significantly improve the speech recognition of Finnish [22] and other highly inflecting and agglutinative languages [23]. We used the Morfessor software [24] for splitting the words. The resulting 460 million tokens in the training set consisted of 8428 unique tokens. The held-out and test sets contained 110 000 and 510 000 tokens, respectively. Full 5-g models were trained for Good–Turing smoothing and for unmodified and modified Kneser–Ney smoothing. The models were pruned to three different size classes: large, medium, and small. SRILM toolkit [25] was used for applying EP to the Good–Turing and the modified Kneser–Ney smoothed models. RKP was performed on the modified Kneser–Ney smoothed model, and KP was performed on the unmodified Kneser–Ney smoothed model. Using KNG, we trained a model to the same size as the full 5-g models and then pruned the grown model with RKP to similar sizes as the other models were pruned to.

The English text corpus was taken from the second edition of the English LDC Gigaword corpus [26]. 930 million words from the New York Times were used. The last segments were excluded from the training set: 200 000 words for the held-out set and 2 million words for the test set. 50 000 most common words were modeled, and the rest were mapped to an unknown word token. Full 4-g models were trained for modified and unmodified Kneser–Ney smoothing. We were unable to train full 4-g models with the SRILM toolkit because of memory constraints, so we used count cutoffs for training a Good–Turing and a modified Kneser–Ney smoothed model to be used with EP. The cutoffs removed all 3-g seen only once and all 4-g seen fewer than three times. With KNG, we trained the largest model we practically could with our implementation. KP was used with the full 4-g unmodified Kneser–Ney model and RKP was used with the full 4-g modified Kneser–Ney model as well as the KNG model. Again, we created models of three different sizes.

The audio data for the Finnish speech recognition experiment was taken from the SPEECON corpus [27]. Only adult speakers in clean recording conditions were used. The training set consisted of 26 h of material by 207 speakers. The development set was 1 h of material by 20 different speakers and evaluation set 1.5 h by set of 31 new speakers. Only full sentences without mispronunciations were used in the development and evaluation sets.

The HUT speech recognizer [28] is based on decision-tree state-clustered hidden Markov triphone models with continuous-density Gaussian mixtures. Each clustered state was additionally associated with a gamma probability density function to model the state durations. The decoder has an efficient time-synchronous, beam-pruned Viterbi token-passing search through a static reentrant lexical prefix tree.

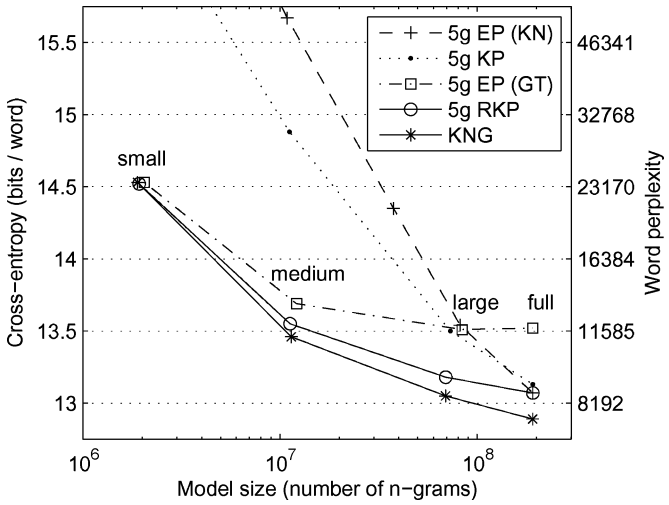

Fig. 3. Cross-entropy results on the Finnish text corpus. Note that the reported cross-entropy and perplexity values are normalized *per word*.

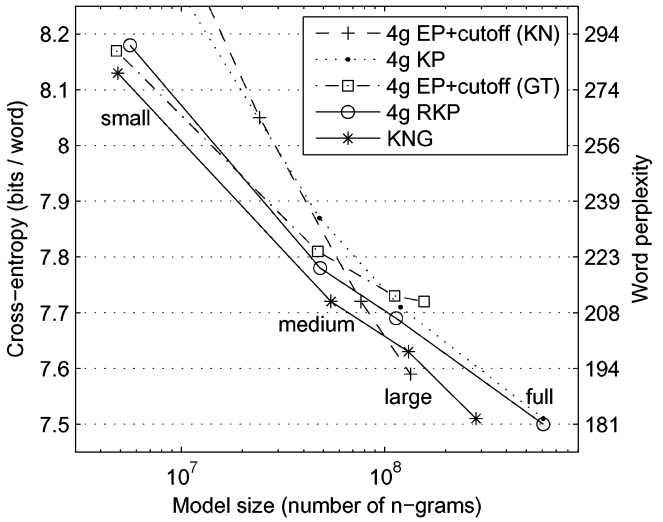

Fig. 4. Cross-entropy results on the English text corpus.

B. Results

For each model M, we computed the cross-entropy H_M with previously unseen text data T containing W_T words

$$H_M(T) = -\frac{1}{W_T} \log_2 P(T|M). \tag{12}$$

The relation to perplexity is $\mathrm{Perp}(T) = 2^{H(T)}$. The cross-entropy and perplexity results for Finnish and English are shown in Figs. 3 and 4. Note that in the Finnish case, the entropy is measured as *bits per word*, and perplexity as *word perplexity* even if the Finnish models operate on subword units. Normalizing entropies and perplexities on whole-word level keeps the values comparable with other studies that might use different word splitting (or no splitting at all). Finnish models were also evaluated on a speech recognition task, and the results are shown in Fig. 5. We report letter error rates (LER) instead of word error rates (WER), since LER provides finer resolution for Finnish

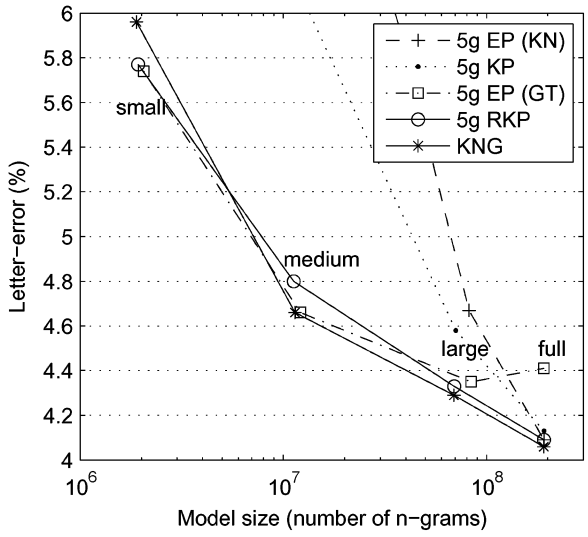

Fig. 5. Results of the Finnish speech recognition task. Note that we report the letter error rate and not the language model token error rate.

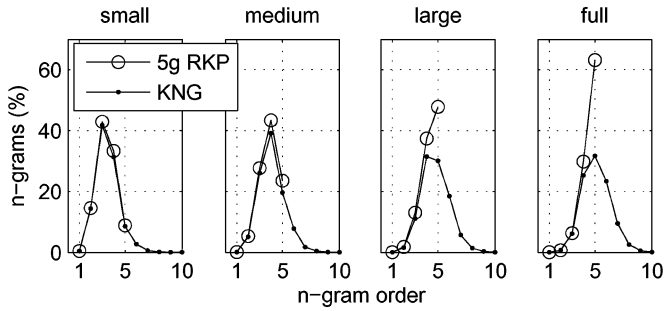

Fig. 6. Distribution of N-grams of different orders in RKP and KNG models for Finnish. Orders up to 10 are shown. The highest order in any model was 16.

words, which are often long because of compound words, inflections, and suffixes. The best obtained LER 4.1% corresponds to WER of 15.1%.

We performed a pairwise one-sided signed-rank Wilcoxon test to see the significance of the differences with $p < 0.01$ to selected pairs of models. In Finnish cross-entropy experiments, the KNG models were significantly better than the RKP models and the entropy pruned Good–Turing models for all but the small models. The RKP model was significantly better than Good–Turing model for all but the small models. In English cross-entropy experiments, all differences between similarly sized Good–Turing, RKP, and KNG models were significant. In Finnish speech recognition tests, the KNG model was not significantly better than the RKP model. The RKP model was significantly better than the Good–Turing model only for the full model.

C. Discussion

In the Finnish cross-entropy results (Fig. 3), we can see that EP and KP degrade the Kneser–Ney-smoothed model rapidly when compared to pruning the Good–Turing-smoothed model. We believe that this is due to two reasons. In Kneser–Ney smoothing, the backoff distributions are optimized for the cases that higher orders do not cover. Thus, the backoff distributions should be modified when N-grams are removed from the model. KP does that, EP does not. However, fixing the backoff distributions does not help if wrong N-grams are removed. Both KP and EP assume that the cost of pruning an N-gram from the model is independent of the other pruning operations performed on the model. This approximation is reasonable for Good–Turing smoothing. In Kneser–Ney smoothing, this is not the case, as the lower order distributions should be corrected to take into account the removal of higher order N-grams.

RKP addresses both of these issues and maintains good performance both for the full Kneser–Ney smoothed model and the grown model. Since the largest KNG model has lower entropy than the full 5-g model, the KNG model must benefit from higher-order N-grams. The advantage is also maintained for the pruned models. Fig. 6 shows how N-grams are distributed on different orders in RKP and KNG models for Finnish. For heavily pruned models, the distributions become almost identical.

Note that for highly inflecting and compounding languages, such as Finnish, the entropy and perplexity values measured on the whole-word level are naturally higher than corresponding English values. This is simply because inflected and compounded words increase the number of distinct word forms. Thus, a Finnish translation typically contains fewer but longer words than the corresponding English sentence.[3] In our test sets, the average number of words per sentence was 11 for Finnish and 20 for English. The sentence entropies for the best models were around 160 bits regardless of the language. Thus, the Finnish word entropy is almost twice the English word entropy, and the perplexity is almost squared.

Also in the English case (Fig. 4), EP and KP seem to degrade results rapidly. Surprisingly, the largest entropy pruned Kneser–Ney model seems to give a good result when compared to other models. That model is actually unpruned, except for count cutoffs. As mentioned in the previous section, count cutoffs were used only for being able to build larger models for EP. The result is in line with [7] where it was reported that count cutoffs can produce better results than plain EP if only light pruning is desired. Preliminary experiments indicate that small cutoffs also improve RKP and KNG.

In speech recognition (Fig. 5), EP and KP degrade the full Kneser–Ney model considerably, too. For example, medium-sized KNG and RKP models have about the same error rate as the large-sized EP and KP models that are almost one order of magnitude larger. Further experiments would be needed for reliably finding out the relative performances of RKP, KNG, and entropy pruned Good–Turing models.

V. Conclusion

This work demonstrated that existing pruning algorithms for N-gram language models contain some approximations that conflict with the state-of-the-art Kneser–Ney smoothing algorithm. We described a new pruning algorithm, which in contrast to the previous algorithms takes Kneser–Ney smoothing into account already when selecting the N-grams to be pruned. We also described an algorithm for building

[3]For example, the six-word sentence *The milk is in the fridge* translates into a three-word sentence in Finnish: *Maito on jääkaapissa*

variable-length Kneser–Ney smoothed models incrementally, which avoids collecting all N-gram counts up to a fixed maximum length. Experiments on Finnish and English text corpora showed that the proposed pruning algorithm gives significantly lower cross-entropies when compared to the previous pruning algorithms, and using the growing algorithm improves the results further. In a Finnish speech recognition task, the proposed algorithms significantly outperformed the previous pruning methods on Kneser–Ney smoothed models. The slight improvement over the entropy pruned Good–Turing smoothed models turned out not to be statistically significant. The software for pruning and growing will be published at http://www.cis.hut.fi/projects/speech/.

REFERENCES

[1] K. Seymore and R. Rosenfeld, "Scalable backoff language models," in *Proc. ICSLP*, 1996, pp. 232–235.

[2] R. Kneser, "Statistical language modeling using a variable context length," in *Proc. ICSLP*, 1996, pp. 494–497.

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

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

[5] J. Goodman, "A bit of progress in language modeling," *Comput. Speech Lang.*, vol. 15, no. 4, pp. 403–434, Oct. 2001.

[6] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proc. ICASSP*, 1995, pp. 181–184.

[7] J. Goodman and J. Gao, "Language model size reduction by pruning and clustering," in *Proc. ICSLP*, 2000, pp. 110–113.

[8] E. Ristad and R. Thomas, "New techniques for context modeling," in *Meeting Assoc. Computat. Ling.*, 1995, pp. 220–227.

[9] M. Siu and M. Ostendorf, "Variable N-grams and extensions for conversational speech language modeling," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 1, pp. 63–75, Jan. 2000.

[10] T. R. Niesler and P. C. Woodland, "Variable-length category n-gram language models," *Comput. Speech Lang.*, vol. 13, no. 1, pp. 99–124, Jan. 1999.

[11] V. Siivola and B. Pellom, "Growing an N-gram model," in *Proc. Interspeech*, 2005, pp. 1309–1312.

[12] H. Yamamoto, S. Isogai, and Y. Sagisaka, "Multi-class composite n-gram language model," *Speech Commun.*, vol. 41, no. 2–3, pp. 369–379, Oct. 2003.

[13] S. Deligne and F. Bimbot, "Inference of variable-length linguistic and acoustic units by multigrams," *Speech Commun.*, vol. 23, no. 3, pp. 223–241, 1997.

[14] M. Creutz and K. Lagus, "Unsupervised discovery of morphemes," in *Proc. Workshop Morphol. Phonol. Learning ACL-02*, 2002, pp. 21–30.

[15] S. Virpioja and M. Kurimo, "Compact n-gram models by incremental growing and clustering of histories," in *Proc. Interspeech*, 2006, pp. 1037–1040.

[16] A. Bonafonte and J. Mariño, "Language modeling using x-grams," in *Proc. ICSLP*, 1996, pp. 394–397.

[17] S. Chen and R. Rosenfeld, "A survey of smoothing techniques for ME models," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 1, pp. 37–50, Jan. 2000.

[18] F. James, Modified Kneser–Ney Smoothing of N-Gram Models Res. Inst. Adv. Comput. Sci., Tech. Rep. 00.07, Oct. 2000.

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

[20] B. Raj and E. W. D. Whittaker, "Lossless compression of language model structure and word identifiers," in *Proc. ICASSP*, 2003, pp. 388–391.

[21] "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. [Online]. Available: http://www.csc.fi/kielipankki/

[22] T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen, "Unlimited vocabulary speech recognition with morph language models applied to Finnish," *Comput. Speech Lang.*, vol. 20, no. 4, pp. 515–541, Oct. 2006.

[23] M. Kurimo, A. Puurula, E. Arisoy, V. Siivola, T. Hirsimäki, J. Pylkkönen, T. Alumae, and M. Saraclar, "Unlimited vocabulary speech recognition for agglutinative languages," in *Proc. HLT-NAACL*, 2006, pp. 487–494.

[24] M. Creutz and K. Lagus, "Unsupervised morpheme segmentation and morphology induction from text corpora Using Morfessor 1.0," Publications Comput. Inf. Sci., Helsinki Univ.Technol., Tech. Rep. A81, 2005.

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

[26] D. Graff, J. Kong, K. Chen, and K. Maeda, *English Gigaword Second Edition*. Philadelphia, PA: Linguistic Data Consortium, 2005.

[27] D. Iskra, B. Grosskopf, K. Marasek, H. van den Heuvel, F. Diehl, and A. Kiessling, "SPEECON—Speech databases for consumer devices: Database specification and validation," in *Proc. LREC*, 2002, pp. 329–333.

[28] J. Pylkkönen, "New pruning criteria for efficient decoding," in *Proc. Interspeech*, 2005, pp. 581–584.

Vesa Siivola received the M.Sc. degree in electrical engineering from the Helsinki University of Technology, Espoo, Finland, in 1999.

Since then, he has been researching language modeling for speech recognition systems in the Adaptive Informatics Research Centre, Helsinki University of Technology.

Teemu Hirsimäki received the M.Sc. degree in computer science from Helsinki University of Technology, Espoo, Finland, in 2002 where he is currently pursuing the Ph.D. degree.

Since 2000, he has worked in the Speech Group, Adaptive Informatics Research Centre, Helsinki University of Technology. His research interests are language modeling and decoding in speech recognition.

Sami Virpioja received the M.Sc. degree in computer science and engineering from Helsinki University of Technology, Espoo, Finland, in 2005.

He is currently a Researcher at the Adaptive Informatics Research Centre, Helsinki University of Technology. His research interests are in statistical language modeling and its applications in speech recognition and machine translation.