Compression Algorithm of Trigram Language Models based on Maximum Likelihood Estimation

Norimichi YODO, Kiyohiro SHIKANO, and Satoshi NAKAMURA
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara, 630-0101 Japan

ABSTRACT
In this paper we propose an algorithm for reducing the size of back-off N-gram models, with less affecting its performance than the traditional cutoff method. The algorithm is based on the Maximum Likelihood (ML) estimation and realizes an N-gram language model with a given number of N-gram probability parameters that minimize the training set perplexity. To confirm the effectiveness of our algorithm, we apply it to trigram and bigram models, and the experiments in terms of perplexity and word error rate in a dictation system are carried out.

1. INTRODUCTION
In large vocabulary continuous speech recognition, N-gram models, which are typical statistical language models, are effective. The bigger N becomes, the higher the ability of N-gram models becomes. However they need a huge number of parameters that grow exponentially with N for the vocabulary size. The huge space of memory results in the system implementation difficulty.

So far several methods have been proposed to reduce the size of N-gram models such as the cutoff method and methods based on information theory. First we overview these techniques and point out their inherent problems. Then we propose an algorithm for reducing the size of a N-gram model. This algorithm is based on the Maximum Likelihood (ML) estimation. When one N-gram parameter is assumed to be removed, we estimate the degradation of language models, namely the maximum likelihood or the perplexity. The smaller the degradation is, the higher priority to be removed the parameter has. Here the ML estimation can realize the N-gram language model with the smallest set of N-gram probability parameters that minimize the training set perplexity. So we can also design an arbitrary size of an N-gram model while keeping the perplexity small.

N-gram models have the so-called sparseness problem that the probability estimation of unknown or rare word sequences is difficult or almost impossible. The back-off smoothing is a popular method which solves this problem. In our algorithm we utilize the heuristics based on the concept of the back-off smoothing. So in this paper we use the back-off N-grams, whose discounting method is the Witten-Bell [1,2].

2. Techniques for Reducing N-gram Parameters
We summarize several techniques that have been proposed so far to reduce the size of N-gram models and point out their problems.

2.1 cutoff
The cutoff method simply excludes from the training data N-grams that occur infrequently and then estimates N-gram parameters. The bigger a training text becomes, the more the number of rare N-grams increases. Just by excluding N-grams with a count of one or two, this method provides large reduction of N-gram probability parameters. However it doesn’t take into consideration two factors as shown below:

1. difference of the value of a probability in original model and the one estimated by the back-off smoothing

2. correlation with other N-grams that have the same context, namely previous N-1 words. For example, is it rational that N-gram parameters that appear once in the training text are similarly excluded when their conditional probabilities differ so much?

2.2 Information Amount
Most of alternative methods are based on information theory, such as the divergence (also called as relative entropy, or Kullback-Liebler distance) of N-gram parameters and (N-1)-gram parameters, or mutual information, where word generation process is regarded as an information source [3,4,5,6]. These methods replace some N-gram parameters with (N-1)-gram parameters at once. Furthermore they don’t take into account the frequency of context, and there are few strict formulations on the relation between the process of parameter reduction and the performance of a language model.

3. ML-based Method
In this section we propose an algorithm of reducing N-gram parameters based on the ML estimation. In comparison with the previous methods, the advantages of our algorithm are as follows:

\begin{itemize}
\item This research is partially supported by the IPA (Information Technology Promotion Agency), JAPAN
\end{itemize}
1. Based on the ML estimation.
   The degradation of a language model can be measured by the ML formulation, and the parameter which has the least degradation is given the top (highest) priority of the N-gram parameter removal.

2. Exclude one parameter which has the top priority, one by one, and an arbitrary size of an N-gram model is available.

The detailed process is as follows:

1. Get the back-off N-gram model and estimate all parameters. (calloff and smoothing methods are optional)

2. Given a context, namely N-1 words, we get two distributions below:
   (1) the conditional probability parameters \{p\} in the original model
   (2) Assuming that one parameter is excluded and estimated by the back-off method, the back-off coefficient needs to be updated to keep the sum of conditional probabilities to 1 (because the real probability values and the estimated values are different in almost all cases).

   Here we get the distribution \{p'\} which differs from the original \{p\} and let \(\alpha'\) be the new back-off coefficient.

3. With the two distributions and the new back-off coefficient \(\alpha'\) above, compute the loss of the entropy (likelihood). The product of the context count and the divergence of the two distributions is proportional to the loss of the entropy, which is the logarithm of the perplexity.

4. Exclude one parameter with the minimum loss of the entropy, or less than a given threshold, and if necessary, update the back-off coefficient to \(\alpha'\).

In the following section we explain the update of the back-off coefficient and the calculation of entropy increase in detail. To make things simple, we use trigram models.

### 3.1 Entropy Increase

Step 2 in the above process gives two distributions \{p\} and \{p'\} as shown in Fig.1. We can now calculate the entropy increase using these two distributions \{p\}, \{p'\} and the updated back-off coefficient \(\alpha'\). \(p()\) represents the conditional probabilities and \(P()\) represents the observed probabilities in the training data, respectively. Let \(c_1\) be a set of trigrams observed in the training data.

We assume that one observed trigram \(w_xw_yw_z\) is excluded (\(p_i = p(w_x|w_yw_z)\)) and estimated by the bigram probability \(q_i = p(w_x|w_y)\). Then the two distributions \{p\}, \{p'\} and the two back-off coefficients \(\alpha, \alpha'\) are defined as follows (see Fig.1):

\[
\{p\} = p(w_x|w_y^y) \quad (1)
\]

\[
\alpha = \frac{1 - \sum_{w_xw_yw_z \in c_1} q_i}{1 - \sum_{w_xw_yw_z \in c_1} q_i} \quad (2)
\]

\[
\{p'\} = \begin{cases} 
 p_i & i \neq k, w_xw_yw_z \in c_1 \\
 \alpha' \cdot q_i & w_xw_yw_z \in c_1 \\
 \alpha' \cdot q_i & \text{otherwise}
\end{cases} \quad (3)
\]

\[
\alpha' = \frac{1 - \sum_{w_xw_yw_z \in c_1} q_i \cdot p_i + p_k}{1 - \sum_{w_xw_yw_z \in c_1} q_i \cdot q_k + q_k} \quad (4)
\]

Then we can get the divergence of \{p\} and \{p'\} from the equation below:

\[
D(p|p') = \sum_i p_i \log \frac{p_i}{p'_i} \quad (5)
\]

Now let \(P(w_x^y)\) be the probability of occurrence of three words \(w_xw_yw_z\), and \(H\) be the entropy of the language models. Then \(H\) can be calculated as below:

\[
H = - \sum_{w_x^y} P(w_x^y) \log P(w_x^y) \quad (6)
\]

This equation is identical to the formulation of the maximum likelihood estimation and can be rewritten as follows:

\[
= - \sum_{w_x} P(w_x^y) \log P(w_x^y) + \sum_{w_x} P(w_x^y)(- \sum_{i} p(w_i|w_x^y) \log p(w_i|w_x^y))
\]
Now let \( p_i \) be the conditional probability \( p(w_x|w_{x-1}w_{x-2}) \), and we make an assumption that the entropy increase in a whole language model is equal to the one in the local space including the parameters that have the same context. Here,

\[
\Delta \mathcal{H} = \mathcal{H} - \mathcal{H} \\
\simeq P(w_x^y)\left(\sum_i p_i \log p_i^y\right) \\
- P(w_x^y)\left(\sum_i p_i \log p_i\right) \\
= P(w_x^y) \sum_i p_i \log \frac{p_i}{p_i^y} \\
= P(w_x^y) \times D(p||p') \\
= \frac{C(w_x^y)}{C(all)} \times D(p||p'), \quad (7)
\]

where \( C(\cdot) \) represents the frequency count of the word sequences and \( C(all) \) means the total occurrence number of word bigrams. The approximation in the second line is derived from the fact that \( p_i \) may be estimated by discounting and differ from the probability estimated by the maximum likelihood estimation. These equations show that the entropy increase is proportional to the product of the context count and the divergence of the two distributions \( \{p\} \) and \( \{p'\} \) (as mentioned above, there are a few assumptions and approximations).

4. EXPERIMENTS AND RESULTS

To confirm the effectiveness of our algorithm, first the perplexity of the language model designed by our ML-based algorithm is compared with the one of the cutoff algorithm. Then word error rates in a dictation system are also compared. In both experiments, we use 4 years' Japanese newspaper corpus[7], 45 months for the training test and 3 months for the test text. The training text comprises 2.3M sentences and 65.3M words (290K unique words)\(^1\).

4.1 Perplexity

We set about 5000 words and 20000 words to vocabulary. They cover 85.8% and 95.7% of words in training text respectively. We compare the compressed trigram models by our algorithm with those by the cutoff. Fig.2 represents the test set perplexity of the trigram models. The trigram models on rightmost points in the graph are initial cutoff condition, 4 and 4 for bigram and trigram respectively. As shown in Table 1, in terms of the perplexity, the trigram models by our method show the same performance with nearly 30% parameters of the language models based on the cutoff at the best point. The size of trigram models are not proportional to the trigram parameters because we keep all bigram parameters. All language models are stored in ARPA format.

Our algorithm is also applied to the bigram models. Fig.3 shows the test set perplexity of the bigram models. When

\(^1\)Here, a 'word' means a morpheme in Japanese

![Figure 2: Number of trigram parameters vs test set perplexity](image)

![Figure 3: Number of bigram parameters vs test set perplexity](image)

<table>
<thead>
<tr>
<th>Table 1: comparison of trigram models</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
</tr>
<tr>
<td>5K</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>20K</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

4.2 Word Error Rates

The experiments of large vocabulary continuous speech recognition are carried out using the Japanese speech dictation system JULIUS[8,9], tripophones (3000 states, 8 mixture)[9] and 200 speech data (100 sentences, male and female) in JNAS (Japanese Newspaper Article Sentences) database[10]. In these experiments we use completely open speech data to the acoustic models and the language mod-
The Japanese dictation system JULIUS adopts 2-paths search, normal left-to-right (LR) bigram based search for the first path and right-to-left (RL) trigram based reverse best-first search for the second path. More than a half of memory in Japanese dictation system JULIUS is occupied by a trigram model. We use the only bigram model at the first path, and various compressed trigram models at the second path.

**Figure 4** shows the recognition experiment results. Our ML-based algorithm can keep the same word error rate between $5 \times 10^4$ and $3 \times 10^5$, which means that our algorithm can reduce trigram parameters into $1/3 \sim 1/5$ compared with the cutoff algorithm.

5. CONCLUSION

This paper proposed a reduction algorithm of the $N$-gram parameters which is based on the maximum likelihood estimation. The algorithm takes into consideration the degradation of language models and realizes arbitrary size of language models. To evaluate the performance of the proposed algorithm, experiments in terms of the perplexity and the word error rate are carried out. In the perplexity, the proposed method realizes language models that show the same value with less parameters than those by traditional cutoff. In large vocabulary continuous speech recognition, not so evident as in the perplexity, the proposed method achieves word error improvement.

As future works, we need to carry out speech recognition experiments using much more data, and adapt the proposed method to variable length $N$-gram models.

---

We can see that in the right small part of the graph, where a large number of trigram parameters are remaining, trigram models by our algorithm is worse than those by cutoff. This is because the amount of speech data is small.

---

REFERENCES


