POS-based Language Models for Large Vocabulary Speech Recognition on Embedded Systems

Petra Witschel, Sergey Astrov, Gabriele Bakencker, Josef G. Bauer, Harald Höge
Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81730 Munich, Germany
Petra.Witschel@siemens.com

Abstract
Speech recognition on embedded systems requires components of low memory footprint and low computational complexity. In this paper a POS-based (part of speech based) language modeling approach is presented which decreases the number of language model parameters combined with a method for reducing memory consumptions via quantization of language model penalties. For the application of short message dictation a language model with about 10,000 words of vocabulary is generated. Using the POS-based language modeling approach the number of parameters comprises 70,058 penalties. The memory consumptions for storing those penalties are reduced about 50% using the presented coding method. Experiments show that the POS-based language model is able to reduce the WER up to 65% for n-best isolated word recognition in comparison to the case without language model. Moreover the increase of WER caused by coding of the language model penalties is not significant.

Keywords: POS-based language models, language models for embedded systems, coding of language model penalties.

1. Introduction
Recently techniques have been developed for ASR to reduce memory footprint of HMMs and computational complexity (see e.g. [1], [3]). Speech recognition for applications based on large vocabulary and on detailed grammatical structure requires complex language models. For using those language models on embedded systems a low memory language modeling approach is needed. Approaches based on word n-grams usually have relatively high memory consumptions for storing transition penalties especially for large vocabulary applications. In this paper a POS-based (part of speech based) language modeling approach is presented which shows the following characteristics:

- The approach is based on a relatively small number of parameters (compared to n-gram language models based on words) as described in sections 2 and 5.2.
- The memory footprint is further reduced via coding of language model penalties (see sections 3 and 5.3).

In section 4 the generation of a language model based on POS classes is described. The application domain of this language model is dictation of short messages. Section 5 presents experiments performed by an isolated word recognition system and results of offline tests. Section 6 concludes with discussion.

2. Language Model
A language model approximates the probability \( P(W) \) for a given word chain \( W = (w_1, ... , w_k) \). Our model is based on POS with their linguistic features and values as found in linguistic lexica (e.g. [5]). Out of these POS clusters are defined which build a set \( C \) of POS classes [8]. Each word belongs to a sub set \( C(w) \) of \( C \). Given the POS classes \( C(w) \) we construct a bigram language model \( P_{LM}(W) \) according to the following equation (see [9]):

\[
P_{LM}(W) = P(w_1) \prod_{i=2}^{k} \left( \sum_{C(w_i)} \sum_{C(w_{i-1})} P(w_i | C(w_{i-1})) \cdot P(C(w_i) | C(w_{i-1})) \right).
\]

The summation over the classes \( C(w_i) \) and \( C(w_{i-1}) \) concerns all POS classes the words \( w_i \) and \( w_{i-1} \) belong to. Language probabilities \( P(w_i | C(w_{i-1})) \) and \( P(C(w_i) | C(w_{i-1})) \) are referred as “word probabilities” and language model probabilities \( P(C(w_i) | C(w_{i-1})) \) as “class bigram probabilities”.

3. Memory Reduction via Quantization of Language Model Penalties
In this section a method for reducing memory consumptions of language models is presented. For this method it is not required to keep the original probability values. A moderate increase of word error rate (WER) of the resulting recognition process is accepted.

For processing on embedded platforms the language model probabilities (see section 2) have to be transferred into negative log domain and quantized to integer values. Language model probabilities \( P(\{\}) \) transformed according to equation (2) are called language model penalties:

\[
Pen(P(\{\})) := (\text{int}_{16})(-\gamma \cdot \log(P(\{\}))) .
\]

The “(\text{int}_{16})” operator converts float values into 16-bit integer numbers. Constant value \( \gamma \) together with the integer operator are aimed to transform the \( \log(P(\{\})) \) in a suited integer value range which fit to the correspondent acoustic
penalties as used in the search process during recognition. The constant $\gamma$ is often denoted as the language modeling factor.

On basis of a training corpus language model penalties as given by (2) are determined leading to $T$ penalties of size of $N$ bytes. The $T$ penalties are clustered into $M$ clusters and each cluster is then mapped into a single “approximation” vector. The set of these vectors is called codebook. By this operation each of the $T$ penalties can only have the values of one of $M$ different codebook vectors of the size of $N$ bytes.

![Figure 1: Clustering language model penalties.](image)

3.1. Finding Clusters

For clustering $M$ clusters have to be defined on interval $[L,R]$ breaking up this interval in sub-intervals. $L$ stands for the value of the lowest penalty and $R$ for the largest penalty value. Then for each cluster one codebook vector has to be found. In order to obtain clusters and corresponding codebook vectors an “equidistant” coding approach is investigated. The interval $[L, R]$ is split up into $M$ intervals $[g_{m-1}, g_m]$ of the same size:

$$g_m - g_{m-1} = \frac{R - L}{M} = \text{const} .$$

(3)

Each vector from cluster $m$ is mapped into a codebook vector $v_m$, which was defined by

$$v_m = \left\lfloor \frac{g_{m-1} + g_m}{2} \right\rfloor .$$

(4)

3.2. Reduction of Memory Consumption

Each penalty that belongs to one cluster is replaced by a pointer, which is the index of the codebook vector for this cluster. It is enough to store the pointers to the corresponding codebook vectors.

Each pointer occupies $\log_2(M)$ bits. The memory assumption of penalties after clustering $\text{mem'}$ (in bytes) consists of the memory consumption of the pointer and of the codebook:

$$\text{mem'} = \frac{T \cdot \log_2(M)}{8} + M \cdot N .$$

(5)

4. Constructing a Language Model for Dictation of Short Messages

4.1. Training Corpora

For estimating language model parameters and defining the vocabulary for dictation of short messages the following three text collections are used:

- collection of text data containing short messages, emails and Usenet texts referred in this paper as “MsgCollection”,
- corpus of the BMBF project VERBMOBIL containing utterances for appointment scheduling (see [7]),
- about 2,500 sentences of “Frankfurter Allgemeine Zeitung” corpus (see [4]).

From the reworked “MsgCollection” and VERBMOBIL data sub corpora are extracted according to statistical criteria. Therefore data are subdivided into sections. A section in the “MsgCollection” e.g. is defined as one contribution to a Usenet group or in “VERBMOBIL” as one dialog step. A table is created, which contains for each section the following information:

- number of sentences,
- average length of sentences,
- number of words of text,
- number of unknown words (relating to the linguistic background lexicon).

A section is selected from the corpus if it fulfills the following criteria. For the sections from “MsgCollection” data those conditions are listed below:

- the section has to contain at most 5 sentences,
- the average sentence length within each section is at most 9 words and the maximum sentence length is at most 14 words,
- at most 0.01 % of the words of the section are unknown.

The whole training corpus called “TrainShortMessage” contains about 370,000 words of text (see figure 2). It is composed of sub corpora of “MsgCollection” (about 140,000 tokens), VERBMOBIL (about 140,000 tokens) and FAZ (about 90,000 tokens).

![Figure 2: Training corpus “TrainShortMessage” containing texts of three different collections.](image)
For generating the language model the POS class set consists of 162 classes. Features and values to define those classes are taken out of a large linguistic lexicon for German (see [5]). Some exemplary features and values are listed in table 1.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>main category</td>
<td>noun, verb, adjective, determiner,...</td>
</tr>
<tr>
<td>number</td>
<td>singular, plural</td>
</tr>
<tr>
<td>person</td>
<td>1st, 2nd, 3rd</td>
</tr>
<tr>
<td>gender</td>
<td>masculine, feminine, neuter</td>
</tr>
<tr>
<td>case</td>
<td>nominative, genitive, dative, accusative</td>
</tr>
<tr>
<td>type of declension</td>
<td>weak, strong</td>
</tr>
<tr>
<td>degree</td>
<td>positive, comparative, superlative</td>
</tr>
</tbody>
</table>

The language model training corpus “TrainShortMessage” is tagged on basis of the vocabulary found in “MsgCollection” data in respecting only words of higher frequency and including predefined lists of special forms like numbers, smilies, month names, day names and punctuations. The size of the resulting vocabulary is 9,796 words.

For testing the language model a test corpus containing short messages has been collected. People were asked to write down meetings and sending congratulations. The test corpus finally consists of 6,503 tokens and 1,616 words of vocabulary.

5. Experimental Results

5.1. Test Data

For testing the language model a test corpus containing short messages has been collected. People were asked to write down short messages for some everyday situations like arranging meetings and sending congratulations. The test corpus finally consists of 6,503 tokens and 1,616 words of vocabulary. A coverage of 89.73% is achieved on basis of the language model vocabulary. For the evaluations the unknown words of the test corpus (e.g. proper names) are added to the language model using an adaptation approach for language models based on POS classes (see [9]).

On basis of the test corpus speech recordings from 23 male and female speakers have been performed. The resulting database was taken to generate n-best lists using an isolated word recognizer.

5.2. Parameters of the Language Model

The bigram language model “ShortMessage.lm” which is based on 162 POS classes and has 9,796 words of vocabulary results in a number of 70,058 parameters inclusive floor values. The training corpus “TrainShortMessage” is large enough to train this language model.

For a corresponding word based bigram language model of the same vocabulary size the number of parameters comprises 9,796² (floor values included). The training corpus “Train-ShortMessage” is much too small to estimate reliably the word bigram probabilities. Most of the word bigrams (95,930,133) are not seen. If we would use a much larger training corpus much more word bigrams could be estimated reliably. The parameters needed for such a word based bigram language model would result in a much higher number than parameters needed for the class based approach.

5.3. Memory Reduction

For language model “ShortMessage.lm” (see section 4.2) there are \( T_w = 26,244 \) penalties of size \( N = 2 \) bytes (for language model class bigram probabilities) and \( T_c = 43,814 \) penalties of size \( N = 2 \) bytes (for language model word probabilities). This results in memory requirements of \( mem \) bytes for the penalties of “ShortMessage.lm” without coding:

\[
\text{mem} = T_w \cdot N + T_c \cdot N = 52488 + 87628 = 140116 \quad (6)
\]

For coding penalties of class bigram probabilities and penalties of word probabilities separate codebooks are used. Each codebook consists of \( M = 256 \) code vectors (8 bit pointers are easy to process). Finally the coded penalties of “ShortMessage.lm” require \( mem' \) bytes of memory:

\[
mem' = \frac{26244 \cdot \log_2(256)}{8} + 256 \cdot 2 + \frac{43814 \cdot \log_2(256)}{8} + 256 \cdot 2 = 71082 \quad (7)
\]

A memory reduction of about 50% is achieved in quantization of language model penalties. The resulting language model is called “ShortMessageCoded.lm”. For recognition results using the coded language model see section 5.4.

5.4. Recognition Results

For the recognition experiments the language model is integrated via a post processing algorithm into the recognition system. On embedded devices like PDAs and mobile phones the recognition result can be presented in form of a result list which contains a limited number of word hypotheses. If the “best” hypothesis (on first position) of the result list is not the expected word the user can select one of the other words. The user may also have the possibility to type-in a new word. Words which are not in the language model vocabulary are then handled as belonging to the “unknown” class of the language model (see section 4.2).
Our isolated word recognizer (see [2]) is based on hidden Markov models with 7,500 Gaussian densities and returns a list of n word hypotheses for each recognition step with corresponding acoustic scores. The pronunciation lexicon contains 9,796 entries plus the pronunciations of the unknown words. During post processing the n-best list is sorted according to combined scores where the word with the minimum combined score is placed on first position. The combined score is calculated as a summation of the acoustic score and the language model score for each word \( w_i \) for

\[
I = 1, \ldots, n
\]

of the n-best list. The language model score is determined on basis of the probability

\[
P(w_i | c_{i-1}^{best})
\]

where \( w_{i-1}^{best} \) and \( c_{i-1}^{best} \) on one hand comes from the previous recognition step as values of lowest combined score. The word \( w_{i-1}^{best} \) (and consequently the appropriate class \( c_{i-1}^{best} \) ) may also have been selected by the user. To simulate the user input in offline evaluations the respective predecessor word \( w_{i-1}^{best} \) (and consequently the appropriate class \( c_{i-1}^{best} \) ) is set to the word given in the transcription of the test corpus.

During the evaluation word error rates (WER) for the non coded language model “ShortMessage.lm” (see section 4.2) and the coded language model “ShortMessageCoded.lm” (see section 5.3) are determined. For comparison also WERs of recognition without a language model are calculated. In the following table WERs are presented respecting the n best recognition results. For example for determining the WER for \( n = 10 \) a word is counted as recognized correctly if it can be found under the first 10 entries of the sorted result list.

<table>
<thead>
<tr>
<th>n</th>
<th>WER with “ShortMess.lm”</th>
<th>WER with “ShortMessCoded.lm”</th>
<th>WER without language model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25.4</td>
<td>25.6</td>
<td>48.8</td>
</tr>
<tr>
<td>2</td>
<td>16.0</td>
<td>16.0</td>
<td>33.2</td>
</tr>
<tr>
<td>3</td>
<td>11.9</td>
<td>12.0</td>
<td>27.0</td>
</tr>
<tr>
<td>6</td>
<td>6.9</td>
<td>7.1</td>
<td>18.2</td>
</tr>
<tr>
<td>10</td>
<td>4.5</td>
<td>4.7</td>
<td>13.3</td>
</tr>
</tbody>
</table>

Table 2: Word error rates (WER) for short message dictation respecting n best recognition hypotheses.

The experiments show a relative increase of WER of less then 2% using the coded language model compared to the non coded language model. Here a reduction of WER up to 65% was achieved comparing n-best recognition without language model and recognition using the coded language model.

6. Conclusion

The paper shows that POS-based language models are suited for implementing speech recognition on embedded systems for a large vocabulary task. The task investigated concerns the dictation of short messages modeled by a vocabulary of about 10,000 words. As shown, POS-based bigrams can be estimated from a small training corpus of about 370,000 word tokens. With this approach a 10,000 word recognition task can be handled by a language model defined by only 70,058 parameters. Evaluations show a word recognition rate of 95.5% respecting the 10 best recognition hypotheses. Further the language model parameters can be coded resulting in a footprint of 71,082 bytes. The coded language model leads to a negligible decrease of word recognition rate to 95.3%. As next step investigations in handling the language model floor values separately for further memory reduction are planed.

7. References