VARIABLE-LENGTH SEQUENCE LANGUAGE MODEL FOR LARGE VOCABULARY CONTINUOUS DICTATION MACHINE
I. Zitouni, J. F. Mari, K. Smaïli, J. P. Haton
LORIA / INRIA-Lorraine
B.P.239 – 54506 Vandœuvre-lès-Nancy, France
E-mail: {zitouni, jfmari, smaili, jph}@loria.fr

ABSTRACT
In natural language, some sequences of words are very frequent. A classical language model, like n-gram, does not adequately take into account such sequences, because it underestimates their probabilities. A better approach consists in modeling word sequences as if they were individual dictionary elements. Sequences are considered as additional entries of the word lexicon, on which language models are computed. In this paper, we present two methods for automatically determining frequent phrases in unlabeled corpora of written sentences. These methods are based on information theoretic criteria which insure a high statistical consistency. Our models reach their local optimum since they minimize the perplexity. One procedure is based only on the n-gram language model to extract word sequences. The second one is based on a class n-gram model trained on 233 classes extracted from the eight grammatical classes of French. Experimental tests, in terms of perplexity and recognition rate, are carried out on a vocabulary of 20000 words and a corpus of 43 million words extracted from the “Le Monde” newspaper. Our models reduce perplexity by more than 20% compared with n-gram (n≤3) and multigram models. In terms of recognition rate, our models outperform n-gram and multigram models.

1. INTRODUCTION
A language can be viewed as a stream of words emitted by a source. Since this language source is subject to syntactic and semantic constraints, words are not independent, and the dependencies are of variable length. One can therefore expect to retrieve, in a corpus, typical variable-length sequences of words, as a means for introducing additional constraints, and hence improving the performance of speech recognizers. In fact, looking at the transcripts we noticed that some sequences of words are very frequent. To take advantage of this fact, we propose to build a language model which bundles sequences of words which are extracted from frequent phrases. The tokens can be both single words and sequences of words. However, introducing word sequences as additional dictionary entries increases the sparseness of the training data and thus deteriorates the quality of the probability estimates for the language model. Therefore, word sequences may not be arbitrarily included in the initial vocabulary.

The n-SeqGram and n-SeqClass models presented in this paper, are aimed at retrieving sequential variable-length regularities within streams of observations by reducing test perplexity. These typical variable-length sequences are automatically extracted from text data. The basic idea is to improve the optimization criterion by making local optimizations.

In section 2, we discuss the principal language models based on variable-length sequences. In section 3, we introduce our approach and give a theoretical background of the n-SeqGram language modeling. An original approach is described in section 4 where classes of words are used instead of words. This elegant method, named n-SeqClass, allows better estimates of the language model. Then, we report in section 5 an evaluation of the language model and the speech recognition system which both use the n-SeqGram and class n-SeqGram (n-SeqClass). In that section we present how variable-length sequences of words are used in our dictation machine MAUD. Finally, we give in section 6 a conclusion and some perspectives.

2. PRINCIPAL VARIABLE-LENGTH SEQUENCE MODELS
One of the most successful language models used in speech recognition is the n-gram model, which assumes that the statistical dependencies between words are of fixed length n. In the interpolated n-gram model [1], the probabilities of the n-grams are estimated by interpolating the relative frequencies of all k-grams, with $k \leq n$, which is a way to account for variable-length dependencies.

Another approach of language modeling assumes that statistical dependencies lie between typical variable-length sequences of words. R.L. Mercer [2] shows that typical sequences of words should be basic elements of the vocabulary, rather than composites taking up all n positions of the n-gram. An automatic way of deciding which sequences should be included in the vocabulary is based on an iterative use of the concept of mutual information between two adjacent words $w_1$ and $w_2$:

$$J(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)} = \log \frac{C(w_1, w_2)N}{C(w_1)C(w_2)} (1)$$

Where C denotes the count function and N the size of the training corpus. A large value of $J(w_1, w_2)$ indicates that $w_1$ and $w_2$ occur as a sequence much more frequently than can be expected from pure chance. $J(w_1, w_2)$ can be used as a measure of desirability of including $w_1$ and $w_2$ as a unit into the vocabulary. Hence, two thresholds, $T_J$ and $T_{\text{min}}$, are established and the basic vocabulary $V_0$ is augmented with all phrases $w_1 w_2$ for which simultaneously
In this way, a new vocabulary $V_j$ is set up. The second threshold $T_{min}$ is needed to make sure that the first comparison of (2) is reliably based on sufficient data. The counts are then adjusted to reflect their new base vocabulary $V_j$. The process is iterated on $V_j$ until no new sequences over the current vocabulary can satisfy (2). In principle, the final vocabulary may obviously include phrases of unlimited length.

Giachin et al. [3][4] suggest to determine the word sequences automatically with an optimization algorithm accruing at reducing test perplexity, $PP$, [5]:

$$PP = 2^{-\sum_{w_i} p(w_i, w_{j}, ..., w_{k})}$$

where $N$ denotes the size of the test corpus and $p(w_i, w_{j}, ..., w_{k})$ the probability on the training corpus.

The basic idea in Giachin’s model, is to classify the vocabulary in $M$ possible classes, in order to make local optimization on a class basis rather than on a word basis and to choose at each iteration the pair that best reduces the log-probability from the training class corpus. This training class corpus is also used to estimate the perplexity which is based on class bigram.

Suhm et al. [6] use the same concept suggested by Giachin et al., to estimate the perplexity which is based on class bigram. Let $J(s_i, s_j)$ be the mutual information of the couple of words or sequences $s_i$ and $s_j$ on the training corpus, and $p$ denotes the coefficient used to compute $T_j$.

Let $T_{min}$ be the minimum count of a candidate sequence. The algorithm is as follows:

1. Determine, on the training corpus, the couples of words or sequences $s_i, s_j$ for which the mutual information $J(s_i, s_j)$ is greater than $T_j$. The total number of words in each couple of words or sequences should be less than $q$;

2. Add the set of new sequences $\{s_i, s_j\}$, deriving from the concatenation of couples $s_i, s_j$ obtained in 1, to the vocabulary and modify the corpus accordingly;

3. Repeat until test perplexity doesn’t decrease.

The core of the algorithm is the computation of the perplexity of the test corpus before and after adding the candidate sequences. The perplexity on the corpus is computed by means of an interpolated n-gram model. It is important to mention that the test corpus should contain sentences which have the same context as those in the training corpus.

The perplexity does not depend on the number of cycles. That’s why the stopping point is not known a priori. Besides, a very high number of new word sequences are usually generated before perplexity begins to increase. Another way to stop the algorithm is to determine when a sufficient number of word sequences has been generated.

It is important to note that, in order to generate long word sequences (e.g., “what time is it”), many shorter sequences have to be generated first (e.g., “what time”). Some of these shorter sequences are no longer useful after the longer ones have been generated, so that they have to be discarded and their original component words should be used instead.

The computation of perplexities is expensive, but it may be sped up by considering that it is not necessary to take into account all the adjacent word pairs in the corpus to compute the new n-gram and perplexity, but only those including new sequences and their
The test corpus (≈ as a test corpus and keep the remaining part as a training corpus. The test and training corpus, respectively, with the set of training corpora used in this approach, are obtained by tagging grammatical classes of the French language [9]. The test and including punctuation, extracted from the

To evaluate the n-SeqClass model we use a set of the more frequent words in the corpora, is used. A comparison in terms of perplexity (cf. Table 1) for bigram, trigram, class multigram, n-SeqClass and n-SeqGram (n∈{2,3}) models shows that our approach outperforms n-gram for n∈{2,3} (186.72 and 155.67 respectively) and class multigram model (158.64). Table 1 shows that the n-SeqClass is better than the n-SeqGram for n∈{2,3}. We think that this result is due principally to the underlying linguistic knowledge introduced by the sequences. These sequences, in addition of their linguistic contribution, allows to cope with the sparseness of data in the training corpus.

### 5.2 Recognition results

The evaluation was done with MAUD [10], which is a continuous dictation system using a stochastic language model.
MAUD works in 4 steps: Gender identification, word lattice generation by means of a Viterbi block algorithm with a bigram model, N-best sentences extraction by means of a beam search in accordance with combined score of the acoustic and the trigram language models, and finally sentence filtering by means of syntactic constraints in order to obtain the best sentence. Each phoneme is modeled by a second order Markov model \([11]\) with 3 states (HMM2) and each word in the vocabulary is represented by the concatenation of the HMM2 phones which compose it. A version of MAUD based on words (vocabulary of 20K words without sequences) has participated in the AUPELF-UREF campaign for French, and came in second place with a word error rate of 32%.

Word sequences produced by class multigram, n-SeqClass or n-SeqGram \([n \in \{2,3\}]\) models are considered as additional entries to the initial vocabulary of 20K words. An optional silence is inserted between the words that belong to the sequence. The 2-SeqGram, respectively the 2-SeqClass, is introduced in the second step of MAUD in order to build the lattice and to extract the N-best sentences (third step) we use the 3-SeqGram, 3-SeqClass respectively.

A summary of recognition results is presented in Table 2. Results are given in terms of word error rate in both word lattice and sentences (results) produced by MAUD. In these experiments, the recognition is done on 50 test sentences extracted arbitrarily from the test sentences delivered by AUPELF-UREF for the evaluation campaign.

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice</td>
<td>10%</td>
<td>7%</td>
<td>6%</td>
<td>5%</td>
</tr>
<tr>
<td>Sentences</td>
<td>29%</td>
<td>28%</td>
<td>24%</td>
<td>21%</td>
</tr>
</tbody>
</table>

Table 2: Word error rate in both word lattice and sentences produced by MAUD. Results are given for the basic model (without word sequences) \(M1\), basic model with sequences proposed by the class multigram model \(M2\), basic model with sequences proposed by the n-SeqGram model \(M3\) and basic model with sequences proposed by the n-SeqClass model \(M4\).

Results show that, in word lattice, the n-SeqClass model (5%) outperforms the basic model (10%), the n-SeqGram model (6%) and the class multigram model (7%). In addition, our experiments show that in word error rate on sentence results, the n-SeqClass model (21%) outperforms basic model (29%), class multigram model (28%) and n-SeqGram model (24%).

6. CONCLUSION AND PERSPECTIVES

Experiments reported here show that the n-SeqClass and n-SeqGram approaches, presented in this paper, are a competitive alternative to the n-gram and multigram models. On our experimental task (for \(n \in \{2,3\}\)) the n-SeqClass and n-SeqGram outperform, in terms of perplexity and recognition rate, the n-gram and multigram models. In addition, these experiments show that the n-SeqClass model outperforms the n-SeqGram in terms of speech word error rate, as well as in terms of perplexity. To better assess the n-SeqClass compared to the n-SeqGram, it is important to reevaluate these two models on a larger corpus. The modeling capability of these two models can be enhanced by new features, like cache and triggers. Another idea to improve perplexity and recognition rate is to combine these models (n-SeqGram and n-SeqClass) with the multigram model. It also seems interesting to investigate the application of this approach to other problems: e.g., the classical natural language processing field, or even in looking for semantic equivalence classes between word sequences in view of tagging concept.

7. REFERENCES


