Variable-Length Sequence Language Model for Large Vocabulary Continuous Dictation Machine

Imed Zitouni, Jean-François Mari, Kamel Smaïli, Jean-Paul Haton

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 ($n \leq 3$) 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:

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

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 $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$ as a unit into the vocabulary. Hence, two thresholds, T_J and T_{min}, are established and the basic vocabulary V_0 is augmented with all phrases $w_1\ w_2$ for which simultaneously

$$J(w_1,w_2) > T_J \text{ and } C(w_1,w_2) > T_{min} \quad (2)$$

In this way, a new vocabulary V_1 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_1. The process is iterated on V_1 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]:

$$PP = 2^{-\frac{1}{N}\log_2 P(w_1,w_2,...,w_N)} \quad (3)$$

where N denotes the size of the test corpus and $P(w_1,w_2,...w_N)$ 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., with the difference that they suggest to choose the class candidates according to their mutual information, instead of probability.

At each iteration, both methods find the best pair according to an objective criterion. This can lead to intractable results when the size of vocabulary increases.

The multigram model, presented by Deligne et al. [7], makes a different assumption to compute sequences of words: Strings of words are assumed to result from the concatenation of variable-length sequences of words, of maximum length q. The likelihood of a string of words is computed by summing up the likelihood values of all possible segmentations of the string into sequences of words. We call L a segmentation of a string W into z sequences of words: $s(1)...s(z)$. The multigram model computes the joint likelihood $P(W,L)$ as the product of the probabilities of the successive sequences, each of them having a maximum length of q:

$$P(W, L) = \prod_{t=1}^{t=z} p(s(t)) \quad (4)$$

Denoting as $\{L\}$ the set of all possible segmentations of W into sequences of words, the likelihood of W is:

$$P(W) = \sum_{L \in \{L\}} P(W, L) \quad (5)$$

The model is thus fully specified by the set of probabilities, $\{p(s(i))\}$, of all the sequences $s(i)$ which can be formed by combining $1,2,...,$ or q words. This model is applied on a test corpus to extract the best sequences of words (sequences with high probabilities $p(s(i))$) added to the base vocabulary. It is important to note that the huge number of possible sequences in a vocabulary of thousands of words requires intensive computation.

3. THE n-SeqGram MODEL

Motivated by the success of Giachin's and Suhm's approaches, our model, called the n-SeqGram model, aims at bundling variable-length sequences of words drawn from a large vocabulary (20000 words).

Our approach is entirely automatic and minimizes the perplexity by making local optimizations. After fixing the maximum length of a word sequence q, the model starts by identifying the set of word sequences obtained by the concatenation of two words or sequences that produce a perplexity reduction. We choose all the candidate sequences whose mutual information is close to the maximum in the corpus and whose count is above a given threshold. Let T_J be the minimum value of the mutual information. T_J is defined as:

$$T_J = (1-p) \max_{s_i \in V, s_j \in V} J(s_i, s_j) \quad (6)$$

where $J(s_i,s_j)$ denotes 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. However, in our approach we discard all shorter sequences that decrease the test perplexity when their original component words are 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

immediate neighbors. By suitably modifying occurrence counts, the necessary amount of computation may be considerably reduced. Note that, though the perplexity is computed on a "shrunk" corpus, when some phrases have been replaced by single symbols (word sequences), it is correct to always keep the original number of words *N* because it is the actual number of words known by the recognizer.

4. THE n-SeqClass MODEL

Considering the success of class based approaches to cope with the sparseness of data in traditional n-gram modeling, we have explored their potential in our approach. The n-SeqClass model is derived from the n-SeqGram model, with the difference that classes are used instead of words.

We proceed as follows: Start by tagging the test and training corpus with a set of *C* classes, where words are partitioned into equivalence classes that are automatically determined [8]. Thus, the n-SeqGram model is used in this class corpus to extract a set of class sequences *C'*. This set of sequences *C'* is made up of the concatenation of the set of initial classes *C* and class sequences obtained by the n-SeqGram procedure. Then, by using the word test corpus and the corresponding class test corpus, which is labeled by the set class sequences *C'*, we extract the corresponding word sequences. Only word sequences with occurrence greater than a predefined coefficient T_{occ} are added to the initial word vocabulary. In this approach, the inter-word transition probability is assumed to depend only on the word classes. In order to compute the perplexity (equation 3) with n-gram model in this approach, the following well known formula is used:

$$P(w_i / w_{i-1}...w_{i-n}) = P(w_i / c_i)P(c_i / c_{i-1}...c_{i-n}) \quad (7)$$

where w_j is classified by c_j.

5. EXPERIMENTAL EVALUATION

To evaluate the n-SeqGram and n-SeqClass models, a set of experiments were carried out on a French corpus ("LeM"). This corpus represents *2* years (*87-88*) of "Le Monde" newspaper and contains *43* million words. In order to have a test corpus with the same context as the training corpus (see section 3), we first extract *10%* of sentences of each paragraph in the "LeM" corpus as a test corpus and keep the remaining part as a training corpus. The test corpus (≈*5* million words) does not appear in the training corpus. A vocabulary of *20000* words, which represents the more frequent words in the corpora, is used.

To evaluate the n-SeqClass model we use a set of *233* classes, including punctuation, extracted from the *8* elementary grammatical classes of the French language [9]. The test and training corpora used in this approach, are obtained by tagging the test and training corpus, respectively, with the set of *233* classes [8]. In our experiments, to estimate the perplexity in the test corpus, we limited the n-SeqGram and n-SeqClass models to $n \in \{2,3\}$. The coefficient *p* used to compute the mutual information threshold T_I (equation 6) is set to *0.2*. The role of this coefficient is to accelerate the process. The number of occurrences T_{min} above which the couple of words or sequences s_i and s_j are not considered for grouping is set to *25*.

A comparison with the n-gram model ($n \in \{2,3\}$) and the multigram model has been carried out in terms of test perplexity and recognition rate. In order to use this large vocabulary (*20000* words) in the multigram model (presented in section 2), we proceed as follows: We start by applying the multigram model in a class corpora used to evaluate the n-SeqClass model. The test and training corpus of classes are labeled automatically by the set of class sequences produced by the multigram model [7]. Then, by using the word training corpus and the corresponding class training corpus, labeled by the set class sequences proposed by the model, we extract the corresponding word sequences. Only word sequences with occurrence greater than a predefined coefficient T_{occ} are added to the initial word vocabulary. In our experiments, for the class multigram model and for the n-SeqClass model, T_{occ} was set to *50*.

5.1 Perplexity results

Perplexity is usually considered to be a performance measure of language models. It is therefore interesting to look at the test perplexity values obtained by the n-SeqGram and n-SeqClass models and compare it to other approaches.

The experiment concerning the interpolated n-gram model gives a test perplexity of *186.72* for the interpolated bigram model and *155.67* for the interpolated trigram model. The n-gram is interpolated with the "back-off " model [1].

q	P1	P2	P3	P4	P5
2	*188.12*	*172.26*	*174.89*	*143.97*	*144.04*
3	*172.24*	*162.46*	*161.24*	*140.43*	*137.17*
5	*161.37*	*151.23*	*149.37*	*137.96*	*133.56*
8	*158.64*	*148.19*	*144.47*	*132.19*	*128.83*
13	*166.78*	*160.13*	*159.09*	*139.73*	*137.48*

Table 1: Test perplexity of the class multigram (P1), 2-SeqGram (P2), 2-SeqClass (P3), 3-SeqGram (P4) and 3-SeqClass (P5) models (*q* denotes the maximum number of words needed in one sequence).

A comparison in terms of perplexity (cf. Table 1) for bigram, trigram, class multigram, n-SeqClass and n-SeqGram ($n \in \{2,3\}$) models shows that our approach outperforms n-gram for $n \in \{2,3\}$ (*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 $n \in \{2,3\}$. 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 *20*K 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 *20*K 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.

	M1	***M2***	***M3***	***M4***
Lattice	*10%*	*7%*	*6%*	*5%*
Sentences	*29%*	*28%*	*24%*	*21%*

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

[1] Katz M. *Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer*. IEEE Trans. ASSP, Vol. 35, N°. 3, pp. 400-401, March 1987.

[2] Jelinek F. *Self-Organized Language Modeling for Speech Recognition*. Readings in Speech Recognition, pp. 450-506. Ed. A. Waibel and K. F. Lee. Morgan Kaufmann Publishers Inc., San Mateo, California, 1989.

[3] Giachin E., Baggia P., and Micca G. *Language Models for Spontaneous Speech Recognition: A Bootstrap Method for Learning Phrase Bigrams*. Proc. ICSLP94, pp. 843-846, Yokohama, 1994.

[4] Giachin E. *Phrase Bigrams for Continuous Speech Recognition*. Proc. ICASSP95, pp. 225-228, 1995.

[5] Bahl L. R., Jelinek F., and Mercer R. L. *A Maximum Likelihood Approach to Continuous Speech Recognition*. IEEE Trans. PAMI, Vol. 5, March 1983.

[6] Suhm B. and Waibel *A. Towards Better Language Models for Spontaneous Speech*. Proc. ICSLP94, pp. 831-834, Yokohama, 1994.

[7] Deligne S. and Bimbot F. *Language Modeling by Variable Length Sequences: Theoretical Formulation and Évaluation of Multigrams*. Proc. ICASSP95, pp. 169-172, 1995.

[8] Smaïli K., Zitouni I., Charpillet F. and Haton J. P. *An Hybrid Language Model for a Continuous Dictation Prototype*. Proc. EUROSPEECH-97, pp. 2723-2726, Rhodes-Greece, 1997.

[9] Smaïli K., Charpillet F., and Haton J. P. *A new algorithm for word classification based on an improved simulated annealing technique*. In 5th International Conference on the Cognitive Science of Natural Language Processing, 1996.

[10] Fohr D, Haton J. P., Mari J. F., Smaïli K., Zitouni I. *MAUD : Un prototype de machine à dicter vocale*. Actes 1ères JST FRANCIL, pp. 25-30, Avigon -France, 1997.

[11] Mari J. F., Haton J. P., Kriouile A. Automatic Word Recognition Based on Second-Order Hidden Markov Models. In IEEE Transactions on Speech and Audio Processing, 2(1), pp. 22-25, 1997.