LANGUAGE MODELING BY VARIABLE LENGTH SEQUENCES : THEORETICAL FORMULATION AND EVALUATION OF MULTIGRAMS

Sabine DELIGNE and Frédéric BIMBOT

Télécom Paris / ENST - Dept Signal, CNRS - URA 820, 46 Rue Barrault, 75634 Paris cedex 13, France, European Union.
e-mail : deligne@sig.enst.fr, bimbot@sig.enst.fr

ABSTRACT

The multigram model assumes that language can be described as the output of a memoryless source that emits variable-length sequences of words. The estimation of the model parameters can be formulated as a Maximum Likelihood estimation problem from incomplete data. We show that estimates of the model parameters can be computed through an iterative Expectation-Maximization algorithm and we describe a forward-backward procedure for its implementation.
We report the results of a systematical evaluation of multigrams for language modeling on the ATIS database. The objective performance measure is the test set perplexity. Our results show that multigrams outperform conventional n-grams for this task.

1. INTRODUCTION

Language can be viewed as a stream of words put out by a source. This source being subject to syntactic and semantic constraints, words are not independent, but the dependencies are of variable length. One can therefore expect to retrieve, in a corpus of text, typical variable-length sequences of words. The multigram model, first presented in [3], aims at modeling these kinds of dependencies. This paper presents the theoretical background of the multigram model (Section 3), as well as the details of its implementation by means of a forward-backward algorithm (Section 4). We also report a systematical evaluation of the model for a task of language modeling, and we compare it with the n-gram model (Section 5).

2. THE MULTIGRAM MODEL

The n-gram model [1] assumes that the statistical dependencies between words are of fixed length n along the whole sentence. In the polygram model [2], 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. The n-multigram model [3] makes a different assumption : under this approach, a sentence is considered as the concatenation of independent variable-length sequences of words, and the likelihood of the sentence is computed as the sum of the individual likelihood corresponding to each possible segmentation.

Let $W = w(1) \cdots w(t) \cdots w(T)$ denote a string of T words, and let L denote a possible segmentation of W into q sequences of words : $s(1) \cdots s(q)$. The n-multigram model computes the joint likelihood $\mathcal{L}(W, L)$ of the corpus W associated to segmentation L as the product of the probabilities of the successive sequences, each of them having a maximum length of n :

$$\mathcal{L}\,(W, L) = \prod_{t=1}^{t=q} p\,(s(t)) \tag{1}$$

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

$$\mathcal{L}_{\mu gr}(W) = \sum_{L \in \{L\}} \mathcal{L}(W, L) \tag{2}$$

The decision-oriented version of the model parses W according to the most likely segmentation, thus yielding the approximation :

$$\mathcal{L}^*_{\mu gr}(W) = \max_{L \in \{L\}} \mathcal{L}(W, L) \tag{3}$$

For instance, with $T = 4$, $n = 3$, $W = abcd$, and by denoting sequence borders with brackets :

$$\mathcal{L}^*_{3-\mu gr}\,(abcd) = \max \left\{ \begin{array}{l} p([a])\,p([bcd]) \\ p([abc])\,p([d]) \\ p([ab])\,p([cd]) \\ p([ab])\,p([c])\,p([d]) \\ p([a])\,p([bc])\,p([d]) \\ p([a])\,p([b])\,p([cd]) \\ p([a])\,p([b])\,p([c])\,p([d]) \end{array} \right\}$$

whereas, more classically :

$$\mathcal{L}_{3-gr}\,(abcd) = p(a)\,p(b|a)\,p(c|ab)\,p(d|bc)$$

3. PARAMETER ESTIMATION

In this section, we derive Maximum Likelihood (ML) estimates of the multigram model parameters. Let $\mathcal{D} = \{s_1; \cdots; s_m\}$ denote a dictionary that contains all the sequences which can be formed by combinations of 1, 2,... up to n words of the language vocabulary. A n-multigram model is fully defined by a set of parameters Θ consisting of the probability of each word sequence $s_i \in \mathcal{D}$:

$$\Theta = (\theta_i)_{i=1}^{m} \quad \text{where } \theta_i = p(s_i) \quad \text{and } \sum_{i=1}^{m} \theta_i = 1$$

An estimation of the set of parameters Θ from a training corpus W can be obtained as a Maximum Likelihood (ML) estimation from incomplete data [4], where the observed data is the string of words W, and the unknown data is the segmentation L underlying the string of words. Thus, iterative ML estimates of Θ can be computed through an EM algorithm. Let $Q(k, k+1)$ be the following auxiliary function at iteration $k+1$:

$$Q(k,k+1) = \sum_{L \in \{L\}} \mathcal{L}(L|W;\Theta^{(k)}) \log \mathcal{L}(W,L;\Theta^{(k+1)}) \quad (4)$$

where $\mathcal{L}(W,L;\Theta^{(k+1)})$ is the joint likelihood computed as in Equation (1) with the parameter estimates at iteration $k+1$, and $\mathcal{L}(L|W;\Theta^{(k)})$ is the conditional likelihood of the segmentation L given W, at iteration k :

$$\mathcal{L}(L|W;\Theta^{(k)}) = \frac{\mathcal{L}(W,L;\Theta^{(k)})}{\mathcal{L}(W;\Theta^{(k)})}$$

the term $\mathcal{L}(W;\Theta^{(k)})$ being computed according to Equation (2).
The reestimation formula of the i^{th} parameter at iteration $(k+1)$, $\theta_i^{(k+1)}$, can be derived directly by maximizing the auxiliary function $Q(k,k+1)$ over $\Theta^{(k+1)}$, under the constraint that all parameters sum up to one. It yields :

$$\theta_i^{(k+1)} = \frac{\sum_{L \in \{L\}} c(s_i,L) \times \mathcal{L}(L|W;\Theta^{(k)})}{\sum_{L \in \{L\}} c(L) \times \mathcal{L}(L|W;\Theta^{(k)})} \quad (5)$$

where $c(s_i,L)$ is the number of occurences of sequence s_i in segmentation L, and $c(L)$ is the total number of sequences in L. Equation (5) shows that the estimate for θ_i is merely a weighted average of the number of occurences of sequence s_i within each segmentation.
Since each iteration improves the model in the sense of increasing the likelihood $\mathcal{L}(W;\Theta^{(k)})$ [4], it eventually converges to a critical point (possibly a local maximum).
A decision-oriented procedure can readily be derived from the reestimation formula : let $L^{*(k)}$ be the most likely segmentation of W at iteration k :

$$L^{*(k)} = \underset{L \in \{L\}}{Argmax} \; \mathcal{L}(L|W;\Theta^{(k)})$$

By setting :

$$\mathcal{L}(L|W;\Theta^{(k)}) = \begin{cases} 1 & \text{if } L = L^{*(k)} \\ 0 & \text{otherwise} \end{cases}$$

the reestimation formula reduces to :

$$\theta_i^{(k+1)} = \frac{c(s_i, L^{*(k)})}{c(L^{*(k)})} \quad (6)$$

The probability of sequence s_i is thus simply reestimated as the relative frequency of the sequence along the best segmentation at iteration k. This procedure parses the corpus according to a ML criterion and can also be used as an alternative way of training, as is done in [3]. The estimation of the multigram parameters through Equation (5) will be refered to as EM training, and the one through Equation (6) as Viterbi training.

4. FORWARD-BACKWARD ALGORITHM

A forward-backward implementation can be used to avoid the explicit search for all segmentations, thus reducing the complexity of the algorithm from $O(2^T)$ to $O(T)$. This forward-backward algorithm relies on the definition of two forward variables, α and γ, and a backward variable β. To a certain extent, a n-multigram model can be thought of as a n-state Ergodic Hidden Markov Model (EHMM) with state i emitting a sequence of length i, and all transition probabilities being equal. Therefore, the forward-backward training algorithm is quite similar to the one used for HMM training. However, the need for a third variable γ arises from the fact that the number of sequences in a string of words depends on the segmentation considered for that string, whereas the number of observations emitted by a classical HMM is constant whatever sequence of states is considered.
Let $W_{(t_1)}^{(t_2)}$ denote the substring of the corpus W between the words of rank t_1 and t_2. We define a variable $\alpha(t)$ as the likelihood of the partial corpus $W_{(1)}^{(t)}$:

$$\alpha(t) = \mathcal{L}(W_{(1)}^{(t)})$$

Since any segmentation of $W_{(1)}^{(t)}$ ends with a sequence of either 1, 2,... or n words, $\alpha(t)$ can be recursively calculated as indicated in Box 1 :

Box 1 : Recursion formula for variable α

for $1 \leq t \leq T$:

$$\alpha(t) = \sum_{l=1}^{n} \alpha(t-l)\, p([w(t-l+1) \cdots w(t)])$$

with $\alpha(0) = 1$ and $\alpha(t) = 0$ for $t < 0$

In the rest of this section, we will use the notation :

$$\alpha_l(t) = \alpha(t-l)\; p([w(t-l+1) \cdots w(t)])$$

which represents the likelihood of $W_{(1)}^{(t)}$ associated to a segmentation, the last sequence of which is of length l.
Similarily, we define a backward variable $\beta(t)$ as the likelihood of the last $(T-t)$ words of the corpus :

$$\beta(t) = \mathcal{L}\,(W_{(t+1)}^{(T)})$$

and we present its recursive formula in Box 2 :

Box 2 : Recursion formula for variable β

for $1 \leq t < T$:

$$\beta(t) = \sum_{l=1}^{n} p([w(t+1) \cdots w(t+l)])\; \beta(t+l)$$

with $\beta(T) = 1$ and $\beta(t) = 0$ for $t > T$

A third variable $\gamma(t)$ is needed, which represents the average number of sequences in a segmentation of $W_{(1)}^{(t)}$. Since

any segmentation ends with a sequence of either 1, 2,... or n words, and since a segmentation of $W_{(1)}^{(t)}$ ending with a sequence of l words has an average number of sequences which is equal to $\gamma(t-l)+1$:

$$\gamma(t) \;=\; \sum_{l=1}^{n} \left(\gamma(t-l) \;+\; 1\right) \frac{\alpha_l(t)}{\alpha(t)} \qquad (7)$$

The quantity $\frac{\alpha_l(t)}{\alpha(t)}$ can be shown to be the likelihood for a segmentation to end with a sequence of l words, given $W_{(1)}^{(t)}$. Thus, we deduce the recursion formula for γ which is given in Box 3 :

Box 3 : Recursion formula for variable γ

for $1 \leq t \leq T$:

$$\gamma(t) \;=\; 1 \;+\; \sum_{l=1}^{n} \gamma(t-l) \;\; \frac{\alpha_l(t)}{\alpha(t)}$$

with $\gamma(0) = 0$ and $\gamma(t) = 0$ for $t < 0$

The parameter reestimation formula (5) can be rewritten as a function of variables α, β and γ at iteration k, as in Box 4.

Box 4 : Parameter Reestimation Formula :

for a sequence s_i of l words,

$$\theta_i^{(k+1)} = \frac{\sum_{t=1}^{T} \alpha_l^{(k)}(t)\; \beta^{(k)}(t)\; \delta^{s_i}_{[w(t-l+1)\cdots w(t)]}}{\beta^{(k)}(0)\; \gamma^{(k)}(T)}$$

where

$$\delta^{s_i}_{[w(t-l+1)\cdots w(t)]} = \begin{cases} 1 \text{ if } [w(t-l+1)\cdots w(t)] = s_i \\ 0 \text{ otherwise} \end{cases}$$

The set of parameters Θ can be initialized with the relative frequencies of all co-occurences of words up to length n in the training corpus. Then Θ is iteratively reestimated as in Box 4 until the training set likelihood does not increase significantly, or with a fixed number of iterations.

5. EVALUATION

In this section, we assess the n-multigram model in the framework of language modeling, on the ATIS database [5], for several values of n, and both training methods described in Section 3, i.e the EM training and the Viterbi training. We compare the multigram model with the conventional n-gram model. Performances are evaluated in terms of perplexity [1] on the test and training sets. The perplexity of the corpus W of size T words is obtained as :

$$\mathcal{PP}(W) \;= 2^{-\frac{1}{T}\log_2 \mathcal{L}(W)}$$

For the n-multigram model, the initialization procedure is common for the EM and the Viterbi trainings. In both cases, all co-occurences of 1, 2,... up to n symbols are used to get initial estimates of the sequence probabilities. However, to avoid overlearning, we found it efficient to discard unfrequent co-occurences, i.e those appearing strictly less than a given number of times c_0. Then, 10 training iterations are performed either as indicated in Equation (5) (EM training) or as described in Equation (6) (Viterbi training). Sequence probabilities falling under a threshold p_0 are set to 0, except those of length 1 which are assigned a minimum probability p_0. After the initialization and each iteration, probabilities are renormalized so that they add up to 1.

During the test phase, the likelihood of the multigram model can be computed using either Equation (2) or Equation (3), yielding two perplexity values respectively noted $\mathcal{PP}$ and $\mathcal{PP}^*$. These quantities can be computed either from the EM estimates or from the Viterbi estimates, which leads altogether to 4 different scores. Since all sequences of length 1 have a minimum probability of p_0, the likelihood of any string of known words can be computed. Unknown words are considered as a single unknown sequence of length 1 with probability p_0, which is added to the dictionary before normalization. The perplexity measure may be biased, depending on the way unknown words are treated. Therefore, beside the conventional perplexity, we also compute the adjusted perplexity, as proposed in [6] by Ueberla.

For the n-gram model, probabilities are estimated as the relative frequencies of the n-grams in the training corpus. A Good-Turing smoothing technique is used to assign a non-zero probability to unseen n-grams made of known words, according to [7] (quoted in [8]). This step is followed by a renormalization of all conditional probabilities. To compute the test likelihood, the conditional probability of any unknown word X is assigned a fixed value p_0. For instance, $P_{4-gr}(X|abc) = p_0$. When X appears in the left context, the history is truncated after X, and the probability of the corresponding n-gram is obtained from the lower order k-gram. For instance, $P_{4-gr}(c|aXb) = P_{2-gr}(c|b)$.

Experiments are carried out on a filtered version of the ATIS database. The training corpus contains more than 10 000 sentences, i.e more than 100 000 words from a vocabulary of about 900 words. City names, month names, day names, airline names, hours and numbers are each replaced by a specific word. The test corpus is another set of 1000 such sentences, i.e about 10 000 words, with 52 occurences of unknown words, among which 40 are distinct.

We report perplexity results for n-gram and n-multigram models, with $1 \leq n \leq 7$. For n-multigrams, the initialization is done with $c_0 = 4$, which is globally the optimal value, but results with $c_0 = 3$ are not significantly different. We set the fixed probability $p_0 \approx 5 \times 10^{-6}$ which is half the probability of a word occuring only once in the training corpus. The Good-Turing occurence number for unseen n-grams depends on n. For $n = 2$, it is approximately equal to 4.5×10^{-3}. For each value of n, we give the perplexity for n-grams, two perplexities ($\mathcal{PP}$ and $\mathcal{PP}^*$) for n-multigrams trained with the EM algorithm and the same two quantities for n-multigrams trained with the Viterbi algorithm. Table 1 gives the test set perplexities, Table 2 the training set perplexities and Table 3 the number of units.

n	1	2	3	4	5	6	7
n-gram model							
$\mathcal{PP}^*$	91.5	17.4	26.6	71.3	150.3	242.2	344.4
n-multigram model : EM training							
$\mathcal{PP}$	91.5	31.6	21.4	17.3	15.9	15.6	15.4
$\mathcal{PP}^*$	91.5	32.5	22.0	17.7	16.3	15.9	15.7
n-multigram model : Viterbi training							
$\mathcal{PP}$	91.5	31.9	21.8	17.4	16.0	15.7	15.4
$\mathcal{PP}^*$	91.5	32.7	22.4	17.8	16.4	16.1	15.7

Table 1: *Perplexity Values on the Test Corpus*

On the test set (Table 1), the optimal n-gram results are obtained with $n = 2$. The dictionary of bigrams contains slightly more than 7000 units and a test set perplexity of 17.4 is achieved. An equivalent performance is obtained with 4-multigrams ($\mathcal{PP} = 17.3$ and $\mathcal{PP}^* = 17.7$), with only 3800 sequences having a non-zero probability. Higher order multigrams provide even lower perplexities : with 7-multigrams, values of $\mathcal{PP} = 15.4$ and $\mathcal{PP}^* = 15.7$ are reached, with about 5000 sequences, i.e less than the number of bigrams. These results show that n-multigrams model better the language of our task, with a lower number of units than conventional n-grams.

The evaluation of the models on the training set (Table 2) shows that multigrams seem to possess a powerful generalization ability.

The comparison of perplexities $\mathcal{PP}$ and $\mathcal{PP}^*$ on both test and training sets indicates that the single best segmentation accounts for most of the corpus likelihood. The use of a Viterbi training instead of an EM training does not have a large impact on the performances, but the Viterbi procedures leads to about 10 % more sequences.

Finally, the ratio between the adjusted perplexity (not reported in the tables) and the perplexity $\mathcal{PP}^*$ for n-grams and n-multigrams is invariably equal to 1.02, which shows that the differences observed between the models are not owed to the way unknown words are dealt with.

We give, in Figure 1, an example of the segmentation obtained on the first 4 sentences in the test corpus, with 5-multigrams trained with the EM method. The segmentations often show interesting correlations with syntactic and semantic groups.

[depart from *city*]
[which airlines] [depart from *city*]
[find the cheapest one-way fare] [from *city* to *city*]
[find] [all] [flights leaving *city* for *city*] [which] [depart]
[before *hour* in the morning]

Figure 1:
Example of a 5-multigram Viterbi Segmentation
First 4 Sentences in the Test Corpus

6. CONCLUSIONS AND PERSPECTIVES

Our experiments show that the multigram approach is a competitive alternative to the n-gram model in terms of language modeling. On our task, n-multigrams models with $n \geq 4$ outperform the best n-gram model (bigrams), though requiring less units.

n	1	2	3	4	5	6	7
n-gram model							
$\mathcal{PP}^*$	80.5	11.0	5.6	4.3	3.8	3.5	3.4
n-multigram model : EM training							
$\mathcal{PP}$	80.5	25.4	16.4	12.5	11.2	10.6	10.3
$\mathcal{PP}^*$	80.5	26.0	16.8	12.7	11.4	10.8	10.5
n-multigram model : Viterbi training							
$\mathcal{PP}$	80.5	25.6	16.6	12.5	11.3	10.7	10.4
$\mathcal{PP}^*$	80.5	26.1	16.9	12.7	11.4	10.9	10.5

Table 2: *Perplexity Values on the Training Corpus*

n=1	n=2	n=3	n=4	n=5	n=6	n=7
n-gram model						
931	7253	16589	24121	27979	28694	27125
n-multigram model : EM training						
931	1753	3180	3800	4315	4580	4859
n-multigram model : Viterbi training						
931	2203	3610	4231	4763	5081	5312

Table 3: *Number of Units in the Dictionary*

Both models could also be used in a single framework, for instance by estimating n-gram models from the sequences provided in a non-supervised manner by a multigram approach. It also seems interesting to investigate the application of the multigram approach to other issues : for instance, in the natural language processing field, for the search for semantic equivalences between word sequences in view of concept tagging, or at the acoustic-phonetic level, for the automatic definition of speech synthesis and recognition units.

7. REFERENCES

[1] F. Jelinek (1990). *Self-organized language modeling for speech recognition*, in Readings in Speech Recognition, pp. 450-506. Ed. A. Waibel and K. F. Lee. Morgan Kaufmann Publishers Inc., San Mateo, California, 1990.

[2] T. Kuhn, H. Niemann, E. G. Schukat-Talamazzini (1994). *Ergodic Hidden Markov Models and Polygrams for language modeling*, Proc. ICASSP 94, vol. 1 , pp. 357-360.

[3] F. Bimbot, R. Pieraccini, E. Levin and B. Atal (1994). *Modèles de Séquences à Horizon Variable : Multigrams*, Proc. XX^{th} JEP, Trégastel (France), June 1994.

[4] A. P. Dempster, N. M. Laird and D. B. Rubin (1977). *Maximum-Likelihood from Incomplete Data via the EM algorithm*, J. Roy. Stat. Soc., vol. 39, n^o 1, pp. 1-38.

[5] MADCOW (1992). *Multi-Data Collection for a Spoken Language Corpus*, Proc. 5^{th} DARPA Workshop on Speech and Natural Language, pp. 7-14.

[6] J. Ueberla (1994). *Analysing a Simple Language Model - Some General Conclusions for Language Models for Speech Recognition*, Computer Speech and Language (April 1994), vol. 8, n^o 2, pp. 153-176.

[7] I. J. Good (1953). *The population frequencies of species and the estimation of population parameters*, Biometrika, n^o 40, pp. 237-264.

[8] K. W. Church, W. A. Gale (1991). *A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of english bigrams*, Computer Speech and Language (January 1991), vol. 5, n^o 1, pp. 19-54.