Language Modeling for Efficient Beam-Search

Marcello Federico, Mauro Cettolo,

Fabio Brugnara and Giuliano Antoniol

IRST-Istituto per la Ricerca Scientifica e Tecnologica

I-38050 Povo (Trento), Italy

Abstract

This paper considers the problems of estimating bigram language models and of efficiently representing them by a finite state network, which can be employed by an hidden Markov model based, beam-search, continuous speech recognizer.

A review of the best known bigram estimation techniques is given together with a description of the original Stacked model. Language model comparisons in terms of perplexity are given for three text corpora with different data sparseness conditions, while speech recognition accuracy tests are presented for a 10,000-word real-time, speaker independent dictation task. The Stacked estimation method compares favorably with the others, by achieving about 93% of word accuracy.

If better language model estimates can improve recognition accuracy, representations better suited to the search algorithm can improve its speed as well. Two static representations of language models are introduced: linear and tree-based. Results show that the latter organization is better exploited by the beam-search algorithm as it provides a 5 times faster response with same word accuracy. Finally, an off-line reduction algorithm is presented that cuts the space requirements of the tree-based topology to about 40%.

The proposed solutions presented here have been successfully employed in a real-time, speaker independent, 10,000-word real-time dictation system for radiological reporting.

1 Introduction

Most current Continuous Speech Recognition (CSR) systems perform the decoding process, at least in a first stage, on a Finite State Network (FSN), representing a bigram Language Model (LM), through a beam-search based algorithm. Hence, successive passes are performed on a reduced search space, e.g. a word graph, and by exploiting a more powerful LM.

This work compares different ways of estimating bigram LMs and of representing them statically by an FSN, which is employed by a Viterbi based beam-search, continuous speech, and speaker independent Hidden Markov Model (HMM) recognizer.

After setting up the general framework of n-gram LMs, the two main computation schemes for bigrams are introduced: backing-off and interpolation. Within the interpolation scheme, several well known bigram estimation methods are introduced together with the original *Stacked* estimation which compares favorably with the best performing ones. Comparisons based on perplexity were performed on three text corpora providing different data sparseness conditions, while speech recognition accuracy tests are presented for a 10,000-word speech recognition task relative to the A.Re.S. (Automatic REporting by Speech) (Angelini *et al.*, 1994a) [1] applicative domain. Recognition tests confirm a relevant difference between "naive" bigram estimation methods and more refined ones,

[1]A.Re.S. is a real-time CSR system for radiological reporting developed at IRST in collaboration with the Radiological Department of S. Chiara Hospital, Trento.

which instead perform very similarly to each other. The best LM achieved a word accuracy of about 93%.

If better bigram estimates can improve beam-search accuracy, a suitable organization of the search space can improve its speed as well. Two different organizations of the search space were investigated: *linear* and *tree-based*. Both representations exploit a factorization of the bigram probability, derived from the interpolation scheme, and differ in that the latter uses a tree representation for both the unigram and bigram distributions.

Results show that the tree-based representation is better suited to the beam-search algorithm because it outperforms the linear one - it is almost 5 times faster, without affecting recognition accuracy. This improvement permits real-time response for the 10,000-word dictation task.

Response time speed-up comes at the cost of increased space requirements. Hence, if the search space is represented statically and the complexity of the recognition task grows, techniques for efficiently representing LMs become necessary. An off-line reduction algorithm is presented that cuts the space requirements of the tree-based topology to about 40%.

The paper is organized as follows. Section 2 introduces the main concepts of stochastic LMs and in particular n-gram LMs. The two main schemes for bigram LMs are presented and several estimation techniques described: Bayesian, adding-1, Good-Turing formula, absolute (or shift) discounting, and linear discounting. In particular, among the linear discounting methods, the original

Stacked estimation algorithm is presented. Section 3 introduces the beam-search algorithm for CSR exploited here. In Section 4 the above two LM representations are described and the reduction process for the tree-based topology is presented. Section 5 provides experimental results concerning the different LM estimations and representations. After Section 6, with conclusions and indications about future work, four appendix sections follow that complete some of the technical aspects discussed in Section 2.

2 Stochastic Language Models

Stochastic LMs are used extensively in many fields: automatic speech recognition, machine translation, spelling correction, text compression, etc (Brown *et al.*, 1994; Kukich, 1992; Witten & Bell, 1991).

The framework of stochastic LMs can be well represented by an information theoretical model. A sequence of words W, generated by a source with probability Pr(W), is transmitted through a channel and transformed into a sequence of observations Y with probability $Pr(Y \mid W)$. For instance, Y could represent the acoustic signal produced by uttering W, the translation of W from Italian to English, or a typewritten version of W, with possible mis-spellings. The problem is decoding the observation Y into the original form W: that is, finding $\hat{W}$ that maximizes the *a-posteriori* probability $Pr(W \mid Y)$. After applying Bayes' rule

$$Pr(W \mid Y) = \frac{Pr(Y \mid W)Pr(W)}{Pr(Y)}$$

and eliminating the constant factor $Pr(Y)^{-1}$, decoding is equivalent to find:

$$\hat{W} = arg \max_{W} Pr(W)Pr(Y \mid W).$$

Usually, the aim of the LM is to supply the decoding algorithm with the probability $Pr(W)$, or a score for it, for every word sequence W. The class of LMs considered in this paper will be introduced now. Let $W = w_1 \ldots w_N$, the probability $Pr(W)$ can be decomposed as

$$Pr(W) = Pr(w_1) \prod_{t=2}^{N} Pr(w_t \mid w_1 \ldots w_{t-1}).$$

The above product considers probabilities that quickly become difficult to estimate. A simplification can be introduced by conditioning the dependence of each word (regardless of t) to the last $n-1$ words:

$$Pr(W) = Pr(w_1 \ldots w_{n-1}) \prod_{t=n}^{N} Pr(w_t \mid w_{t-n+1} \ldots w_{t-1}).$$

This *n-gram* approximation, which formally assumes a *time-invariant* Markov process (Cover & Thomas, 1991), greatly reduces the statistics to be collected in order to compute Pr(W), clearly at the expense of precision. However, even a 3-gram (trigram) model may require a large amount of data (texts) for reliably estimating a large number of paramenters - e.g. a trigram LM with a vocabulary of 1,000 words requires estimating about 10^9 parameters. Another important aspect that renders n-gram estimation a difficult task is the inherent data sparseness of real texts. Experimentally most of the correct word sequences can be considered rare events, as they generally occur only a few times, if ever, even in very large text collections (corpora). In the next sections, trainable n-gram models will be introduced that try to cope with these two problems: the estimation of a large number of parameters and the data sparseness of texts.

Another important aspect regarding LMs concerns their assessment. In general, LMs are evaluated with respect to their capability of predicting words inside a text. The most used performance measure is the so called *perplexity*. Perplexity is based on the following statistic:

$$\hat{H}_{LM} = -\frac{\log Pr_{LM}(W)}{N}$$

where $W = w_1 \dots w_N$ is a sufficiently long test sequence of the source and Pr_{LM} is the probability of W computed by the LM. It can be shown (Cover & Thomas, 1991) that $\hat{H}_{LM}$ is a *consistent* estimator of the source *cross entropy*:

$$H_{LM} = \lim_{|N| \to \infty} -\frac{1}{N} \sum_{W} Pr(W) \ \log Pr_{LM}(W)$$

assuming that the Markov process obeys the law of large numbers. Hence, the perplexity of LM is defined as:

$$PP_{LM} = 2^{\hat{H}_{LM}}.$$

According to basic information theory principles, perplexity explains the prediction task of the LM to be as difficult as predicting one of PP_{LM} equally likely words. The requirement of W being a test sequence (i.e. not used for training the LM) permits the computation of a more correct statistic as unseen n-grams are also included according to the data sparseness characteristics of the source.

(TABLE I ABOUT HERE)

2.1 Basic Bigram Estimation Methods

In the following, n-grams with $n = 2$ (bigrams) will be considered. In fact, bigrams are still the most popular LM in speech recognition both in single-pass systems and in the first stage of multi-pass systems. However, all the following bigram schemes and estimation methods can be recursively extended to higher-order n-grams with little effort. Given a bigram yz, the LMs treated here require the estimation of the basic bigram probability $Pr(z \mid y)$ from a training sample W. In general, the above probability is computed by combining two components: a discounting function and a redistribution function. The first function is related to the zero-frequency estimation problem (Witten & Bell, 1991): that is, a probability for all the bigrams that never occurred in W is computed by discounting the bigram relative frequency $f(z \mid y) = \frac{c(yz)}{c(y)}$. The second function redistributes the zero-frequency probability among the unseen bigrams. In general, probability is redistributed either according to a less specific distribution - e.g. the bigram distribution if trigrams are computed - or otherwise (e.g. for unigrams) uniformly. The discounting and the redistribution functions are generally combined according to two main schemes: *backing-off* (Katz, 1987) and *interpolation* (Jelinek & Mercer, 1980).

In the backing-off scheme the bigram probability is computed by choosing the most significant approximation according to the frequency countings:

$$Pr(z \mid y) = \begin{cases} f^*(z \mid y) & \text{if } c(yz) > 0 \\ \\ K_y \lambda(y) Pr(z) & \text{otherwise} \end{cases} \tag{1}$$

where $f^*(z \mid y)$ denotes the discounted frequency distribution, $\lambda(y)$ is the zero-frequency probability, and K_y is an appropriate normalization constant such that:

$$0 \leq f^*(z \mid y) \leq f(z \mid y)$$

$$\sum_z f^*(z \mid y) = 1 - \lambda(y)$$

$$\sum_z Pr(z \mid y) = 1.$$

In the interpolation scheme the n-gram probability is computed as a weighted sum of the discounted frequency and the redistributed zero-frequency probability:

$$Pr(z \mid y) = \begin{cases} f^*(z \mid y) + \lambda(y) Pr(z) & \text{if } c(y) > 0 \\ \\ Pr(z) & \text{otherwise} \end{cases} \tag{2}$$

In this work emphasis will be given to the latter scheme as it provides very efficient LM representations for speech recognition (see Section 4) and equivalent, if not superior, performance.

Several frequency discounting methods as well as zero-frequency estimators have been proposed in the literature of information theory, statistics, pattern recognition, speech recognition, etc. Four of the best known techniques have been considered here: the "Adding-1" formula, the Good-Turing formula, absolute discounting, and linear discounting. In Table II a compendium of these approaches is given.

Adding-1 (A1). This very simple estimator results from the Bayesian estimation criterion (Vapnik, 1982) discussed in Appendix A. This method simply adds a constant 1 to all the bigram counts and assigns a probability in proportion to their number to all the never seen events. This estimator tends to over-estimate the zero-frequency probability in presence of very sparse data.

Good-Turing (GT) formula (Good, 1953). The GT formula can be derived (see Appendix B) by assuming the "symmetry" requirement -i.e. same frequencies correspond to equal probability estimates. It must be noted that the GT formula is indeed applied to the bigram counting function:

$$c^*(yz) = (c(yz) + 1)\frac{d_{c(yz)+1}}{d_{c(yz)}}.$$

and that discounting on $f(z \mid y)$ occurs in proportion to $\frac{c^*(yz)}{c(yz)}$. The GT estimator, introduced for backing-off n-gram estimation (Katz, 1987), must be used carefully as there could easily be frequencies r, especially high frequencies, for which d_r results equal to zero. The solution suggested by Katz is to compute a new discounting function h' such that $h' = 1$ for bigrams occurring more than k (e.g. $k = 5$) times, while h' is linear in h for the other bigrams, and it provides the same $\lambda(y)$ values.

Absolute (or "shift") discounting (Ney, Essen & Kneser, 1994; Witten & Bell, 1991). Absolute or "shift" discounting methods subtract a small constant β from all non-zero bigram counts. In the first proposed method (**S1**) all singletons are deleted from the countings and treated as if they were novel events ($\beta = 1$). In this case the backing-off condition in (1) $c(yz) > 0$ must be changed to $c(yz) > 1$. The zero-frequency probability becomes directly proportional to the number of different words occurring after the context y. Experimentally, this method provides lower zero-frequency probabilities than the "adding-one" method. Another advantage is that a significantly smaller amount of n-grams have to be kept in storage, as most n-grams in real texts occur once or twice.

By assuming instead $0 < \beta < 1$ and by applying the Leaving-one-out (LOO) estimation criterion, a different solution (**S**β) was provided by Ney *et al.* (1994) (see Table II). The same authors also claimed that no improvements were seen by assuming β as a function of the context y. It is easy to see that with this solution a smaller amount of probability is assigned to unseen events with respect to the S1 method.

Linear discounting. Empirical frequencies are discounted in proportion to their value. The first estimation method considered, here called Linear Simple

(**LS**), assumes a simple discounting constant that can be estimated by assuming either a Poisson process for new words occurring after a given context (Witten & Bell, 1991), or by applying the LOO estimation method (Ney *et al.*, 1994; Nadas, 1985). In both cases a good approximation of the so computed estimators yields the GT estimator for novel bigrams:

$$c^*(yz)\frac{d_0}{c} = \frac{d_1}{d_0}\,\frac{d_0}{c} = \frac{d_1}{c}.$$

Another discounting method, here called Linear Empirical (**LE**), was described by Witten & Bell (1991) and recently employed for LM estimation (Placeway, Schwartz, Fung & Nguyen, 1993). The basic idea is to make the zero-frequency probability $\lambda(y)$ proportional to the number of "new events" occurring after context y during the production of the text sample.

Finally, the Linear General (**LG**) formula provides the most general linear discounting model. The estimation of the $|V|$ parameters λ, for an interpolated LM model (2), will be described in the next subsection.

(TABLE II ABOUT HERE)

2.2 LG Stacked Estimation

The interpolation scheme, which provides comparable, if not better, results than the backing-off scheme (in terms of PP), permits very efficient LM representations for speech recognition (Section 4). In this section, a particular estimation method for the LG interpolation model is presented that compares favorably with those reported in the literature.

In the LG interpolated model, n-gram conditional frequencies are smoothed with the lower order frequencies. The basic bigram probability is expressed as follows:

$$Pr(z \mid y) = (1 - \lambda(y))f(z \mid y) + \lambda(y)Pr(z) \tag{3}$$

where $0 < \lambda(y) \leq 1 \ \forall y$ and $\lambda(y) = 1$ if $c(y) = 0$. According to the literature (Derouault & Merialdo, 1986; Jelinek, Mercer & Roukos, 1992), estimation of conditional frequencies needs simple counting over a text sample W_f, while interpolation parameters are estimated on a second disjoint training sequence W_λ by means of the following ML iterative estimator (Baum, 1972):

$$\lambda^{n+1}(y) = \frac{1}{|S_y|} \sum_{yz \in S_y} \frac{\lambda^n(y)Pr(z)}{(1 - \lambda^n(y))f(z \mid y) + \lambda^n(y)Pr(z)} \quad \forall y \in V \tag{4}$$

where S_y is the set of all occurrences of bigrams of type $y\_$ in W_λ. Moreover, to avoid overfitting parameters, training can be supervised by evaluating performance of the LM on a small *cross-validation* sequence W_{cv} after each iteration. The drawback of this procedure is that the available data must be split into three disjoint training samples W_f, W_λ and W_{cv}, which requires a lot of data. Improvements can be introduced by employing the Delete Estimation or the LOO method. This leads to maximizing the following log-likelihood:

$$LL = \sum_{y} \sum_{yz \in S_y} log\left((1 - \lambda(y))f^*(z \mid y) + \lambda(y)Pr(z)\right)$$

where $f^*(z \mid y)$ is the relative frequency computed on W after deleting an occurrence of yz. Maximization can be computed by (4) after replacing $f(z \mid y)$ with $f^*(z \mid y)$ (see Appendix D for details).

Even if this method does not require one training sequence, namely W_λ, it still requires some strategy to avoid overtraining parameters. One simple low cost technique is to stop iterations on one parameter as soon as its value stops changing significantly (LG LOO estimation.) On the other hand, experiments showed that a cross-validation sample W_{cv} is more advisable. This sample can be obtained by deleting a random subsequence W_{cv} from the training sequence W. With a cross-validation sample, iterations on each parameter $\lambda(y)$ are stopped as soon as its contribution to the likelihood of W_{cv} worsens.

An interesting way to reduce the disadvantage of deleting a cross-validation set from the training data is provided by the *stacked* version of the LG model (Federico, 1993).

The stacked method (Breiman, 1992) basically combines different predictors estimated on the training data to improve prediction accuracy. Translated into LMs, given m bigram estimates:

$$Pr^1(z \mid y), Pr^2(z \mid y), \ldots, Pr^m(z \mid y)$$

computed on a training sample (level zero data), compute a new LM by linearly combining these estimates (level one data):

$$\sum_{i=1}^{m} \alpha_i Pr^i(z \mid y) : \forall i \ \ \alpha_i \geq 0 \ \text{ and } \ \sum_{i=1}^{m} \alpha_i = 1$$

According to the LG stacked training algorithm, each LM is estimated on a different random partition of the training data W into two sets of bigrams: a frequency and parameter estimation sample $W_{f,\lambda}$, and a cross-validation sample W_{cv}. Parameters are iteratively estimated by the LOO version of formula (4) on $W_{f,\lambda}$ and iterations on single parameters are stopped as soon as their contribution to the likelihood of W_{cv} worsens. After parameter training, frequencies are re-computed on W. When m of such LMs have been estimated they are stacked together by a convex combination. The simplest combination is devised by taking the average of the models. Hence, the following bigram LM results:

$$\frac{1}{m}\sum_{i=1}^{m} Pr^i(z \mid y; f, \lambda^i) = \frac{1}{m}\sum_{i=1}^{m}(1-\lambda^i(y))f(z \mid y) + \frac{1}{m}\sum_{i=1}^{m}\lambda^i(y)Pr(z).$$

The mixture of LMs guarantees to improve PP with respect to the average performance of the single LMs. This is explained by the following property which uses an equivalent but more convenient definition of perplexity:

Property 1 For any convex combination of m bigram LMs and for every text sequence the following inequality holds:

$$\sum_{i=1}^{m} \alpha_i \left(\prod_{t=2}^{N} Pr^i(w_t \mid w_{t-1}) \right)^{-\frac{1}{N-1}} \geq \left(\prod_{t=2}^{N} \left(\sum_{i=1}^{m} \alpha_i Pr^i(w_t \mid w_{t-1}) \right) \right)^{-\frac{1}{N-1}} \quad (5)$$

A proof of this inequality is given in Appendix C. Experimental evidence shows that improvement is also achieved with respect to the best performing LM of the combination.

3 Beam-search

The network representation of an LM is a convenient way of integrating constraints in an HMM based recognizer, because HMMs themselves have a graph structure. If a network is available that expresses LM constraints in terms of word sequence probability, it is easy to expand it by substituting word-labeled arcs with corresponding sequences of unit-labeled arcs, thus building a unit-based LM, which can be searched with the Viterbi algorithm.

The Viterbi algorithm requires that at every time instant, all possible path extensions are considered in order to update the scores of the network states for the next time instant, which can be a very time consuming operation if the network is large. On the other hand, it often happens that only a small fraction of the network states have a high probability, while a large number of states have a probability that compares very unfavorably with the best one, making it very unlikely that they will be visited by the globally optimal path. The beam search method (Ney, Mergel, Noll & Paeseler, 1992) exploits this fact to reduce the amount of computation needed to find the optimal path. While decoding a string, at every time instant the probability of the best path so far is computed, and only those states are expanded whose probability exceeds the value given by this probability times a predefined quantity. Discarding paths on the basis of such a simple criterion may obviously lead to the loss of the solution, but in practice this rarely happens, while the number of states explored during search is dramatically reduced. Moreover, by varying the threshold level it is possible to find a good compromise between accuracy loss and speed gain.

The pruning criterion being based on a comparison between the probability of a state and the probability of the best state, beam search becomes less effective when a lot of states have probability similar to the best one. It turns out that this problem can be alleviated by adopting suitable methods for mapping LM into networks. A tree based representation is a well known approach for doing this, and is adopted in this work which proposes network optimization as a possibility for overcoming the memory requirement problem raised by this kind of organization.

4 LM Representation

In speech recognition, searching is performed by matching the acoustic data with knowledge represented through a finite state network, which contains acoustic and linguistic constraints. Acoustic constraints are given by allowing only those paths representing phoneme sequences that correspond to word transcriptions. Linguistic constraints are imposed by associating bigram LM probabilities to word pairs.

If better bigram estimates can improve the search engine accuracy, a suitable organization of the search space can improve its speed as well, as will be shown in the following subsections.

4.1 Linear Representation

An interpolated bigram LM, expressed by:

$$Pr(z \mid y) = f^*(z \mid y) + \lambda(y)Pr(z)$$

with $0 \leq \lambda(y) \leq 1$ and $\lambda(y) = 1$ when $c(y) = 0$, could be implemented with the explicit representation of all the possible links between word pairs.

Placeway *et al.* (1993) showed how to represent only links between word pairs that were seen within the training data by using a null node for the unseen bigrams (see Figure 1). Each word x is linked to the null node by an arc with probability $\lambda(x)$. This represents the probability of the unseen events in context x. Arcs go from the null node to each initial state of a generic word z with the unigram probability $Pr(z)$. Thus, for each pair (x, z) there is either the straightforward link with which the probability estimated by (3) is associated, when the pair was seen within the training data, or, at least, the two-links path through the null node, when (x, z) was not seen. Note that if a pair was observed, both paths connecting the two words exist, but the one-arc path is more probable since $f^*(z \mid y) > 0$.

(FIGURE 1 ABOUT HERE)

By indicating with V the size of the vocabulary and with d the number of different observed events, only $d + 2|V|$ links connecting words are necessary if the null node is used. For large vocabularies ($|V| > 1000$), this number is much less than the $|V|^2$ required by the fully connected representation.

4.2 Tree-based Representation

The acoustic "similarity" of words is not taken into account in the linear representation. As in a large vocabulary many words share the initial portion of their phonetic transcription, the lexicon can be represented as a tree in which common beginning phonemes of words are shared and each leaf corresponds to a word.

Researchers at Philips laboratories reported advantages obtained by integrating tree organization of the lexicon with the beam-search algorithm (Ney, Haeb-Umbach, Tran & Oerder, 1992; Ney, 1993). They showed that 95% of the state hypotheses were in the first two phonemes of words when the linear representation is used.

This fact makes tree organization attractive since it may prevent repetition of the same computations for active words that share the initial phoneme transcription. Moreover, tree organization would permit savings on used memory space with respect to linear organization. As an example, for the 10000-word dictation A.Re.S. system, the ratio between the number of links required by linear representation of the lexicon and tree organization is 2.69.

Unfortunately, unlike linear representation, in the lexicon tree the identity of a word is only known at the leaf level: so, in order to integrate the bigram probability, a duplicate of the whole lexicon is necessary for each word and the LM probability is applied at the end of the second word of each pair. Such a network would require an unmanageable amount of memory, corresponding to $450 \cdot 10^6$ phoneme labeled links for the A.Re.S. system.

However, the computational efficiency provided by tree-based representation justifies the efforts made by many research laboratories to overcome the problem of its memory requirements.

(FIGURE 2 ABOUT HERE)

4.3 Static Tree-based Representation

To organize the search space in tree mode, it has been proposed either to dynamically build the portion of currently explored space (Ney *et al.*, 1992; Odell, Valtchev, Woodland & Young, 1994), or to adopt a static linear-tree mixture approach (Murveit, Monaco, Digalakis & Butzberger, 1994).

Nevertheless, a static representation of the whole search space is attractive mainly for two reasons: first, there is no overhead in building it during the recognition process; and secondly, the network can be reduced off-line.

A novel network compression scheme is now presented. The idea is based on also using the null node for the tree-based representation (see Figure 2). For each word, the set of successor words that were seen in the training data is organized as a tree. If y is an observed successor word of x, then the probability $P(y \mid x)$ is assigned to the arc connecting the leaf corresponding to y of the successor tree of x with the root of the successor tree of y. The null node is the root of the whole lexicon tree. Each leaf of the lexicon tree corresponds to one of the $|V|$ words of the vocabulary and it is linked to the root of its successor tree by an arc with the corresponding unigram probability $Pr(y)$. Another arc, with probability $\lambda(y)$, links the root of the successor tree of y with the null node. Within every tree, when word transcriptions are prefixes of other ones, their last phoneme-arc is duplicated in order to have different word-end leaves.

In Figure 2 the left-most triangle represents the whole lexicon tree, while small triangles represent successor trees ($st(\cdot)$).

Using successor trees instead of $|V|$ repetitions of the whole lexicon tree, the static representation of the tree-organized network becomes effective. In fact, the average size of successor trees depends on the number of observed bigrams, and is usually much smaller than the size of the whole lexicon. Successor trees can be further reduced by considering only a subset of the word pairs that were seen during the training phase. Some techniques for choosing subsets of bigrams are explained and compared in Murveit *et al.* (1994).

4.4 Factorization of Probabilities

Acoustic information, related to phoneme HMMs within trees, and linguistic information, specified by the bigram probabilities of arcs among trees, are placed in different regions of the tree-based network.

During the beam-search decoding, the sooner this twofold information is available the sooner unlikely hypotheses will be discarded from the beam. Linguistic information provided by the bigram probabilities, weighting arcs outgoing from trees, can be used in advance by factorizing them within the trees. When more words share a phoneme-arc, both in the lexicon tree and in successor trees the upper-bound among their probabilities can be used (see Figure 4).

The algorithm summarized in Figure 3 performs factorization of probabilities in a tree. It requires that probabilities of all arcs be equal to one, except those of incoming-leaf arcs.

(FIGURE 3 ABOUT HERE)

Note that using upper-bounds is not an approximation. The correct LM probability is assigned to a word candidate as soon as the word ceases to share phoneme-arcs with other words.

Using LM scores before word-end leaves in a tree-based representation for pruning purposes has been already exploited (Odell *et al.*, 1994; Steinbiss, Tran & Ney, 1994). In these papers, however, it was specified that the values were calculated on-the-fly during recognition. In the second work, to avoid run-time overload, the use of unigram upper-bounds was also proposed. On the contrary, the factorization of bigram probabilities considered here is performed off-line. This results in no run-time overload and in the application of correct upper-bounds.

(FIGURE 4 ABOUT HERE)

4.5 Tree-based Network Reduction

After factorization of the probabilities, empty arcs outgoing from leaves with probability equal to one can be eliminated by collapsing states linked by them. Moreover, many arcs of ending portions of words (Figure 4) still have an associated probability equal to one. This means that in the tree-based network there are still redundant paths that can be merged to reduce network size.

In the following sub-sections, two possible network reduction algorithms will be discussed.

4.5.1 Optimization

A first possibility is to optimize the network by using one of the available algorithms for minimizing the number of states in a deterministic finite state automaton. These algorithms are based on the *indistinguishability* property of states (Linz, 1990). Let T_1 and T_2 be the automata derived from the initial one by considering as initial states respectively S_1 and S_2. S_1 and S_2 are said indistinguishable if the languages accepted by T_1 and T_2 are exactly the same.

The partitioning algorithm (Aho, Hopcroft & Ullman, 1974) can be used for this purpose. It needs an initial partition of the automaton states and, iteratively, refines the partition by considering the set of states whose next state, given a symbol, is in one particular block of the current partition. Each time a block is partitioned, the smaller generated sub-block is used for further refining. The algorithm ends when, for each given ($block$, $symbol$) pair, all states that point to some state of the given block with an arc labeled with the given symbol, are in the same block. Each block of the resulting partition consists of states that are indistinguishable and can therefore be merged: the so obtained automaton has the minimum number of states among those equivalent to the starting one.

(FIGURE 5 ABOUT HERE)

The algorithm is given in Figure 5. The starting partition can be obtained by partitioning the states into final and non-final states. The time complexity is $O(m\ n \log n)$, where m is the number of different symbols labeling arcs, and n the number of states of the starting network.

Note that symbols of the network of Figure 4 are pairs $(phoneme, probability)$: their number can be very large according to the network structure.

4.5.2 Subtree Isomorphism

The state optimization algorithm gives the greatest reduction of network size, but it does not take advantage of the particular topology of the network.

An elegant way to exploit the tree topology is to use a quadratic algorithm that is still based on the indistinguishability property of states. The algorithm simply considers all possible pairs of states and marks those that result indistinguishable. At the end, such marked states are merged together and the optimum automaton, with respect to the number of states, is obtained.

If the starting automaton is a tree, the indistinguishability property coincides with that of *subtree isomorphism.* In fact, in this case, two states are indistinguishable if and only if their subtrees are isomorphic.

To apply this idea to the problem of reducing the size of the tree-based network, a tree is built by starting from a new root node and connecting it to the roots of all the successor trees and to the lexicon tree after the probability factorization. The new connecting arcs are then labeled with new symbols that are not already used within trees. Then, the algorithm depicted in Figure 6 is used by considering only non-leaves (internal) nodes, in such a way that the final linking can be done according to the bigram LM.

(FIGURE 6 ABOUT HERE)

To reduce the number of isomorphism checks, it is convenient to introduce some additional knowledge about a state, such as the length of the longest path from it to a leaf node. In any case, the time complexity of the algorithm remains $O(n^2)$.

5 Experiments

5.1 LM Comparison on Perplexity

Performance of each LM was evaluated on three text corpora presenting different data sparseness conditions. Detailed figures about each corpus are given in Table III. Among them the *coverage-rate* needs some explanation. This quantifies the percentage of bigram occurrences inside the testing sample that occur at least once inside the training sample. In other words, the coverage-rate tells how often an LM has to rely on the zero-frequency and unigram probabilities. In fact, this gives an idea of the data sparseness conditions of each task. A brief description of the content of each corpus follows.

(TABLE III ABOUT HERE)

The first corpus considered contains transcriptions of roughly 3,000 database queries produced by 53 subjects and collected by means of Wizard of Oz simulations (Corazza, Federico, Gretter & Lazzari, 1993). The application domain is an information desk (called "concierge") able to answer spoken queries concerning IRST's organization, researchers' interests, publication lists, etc. This task was included in the experiments as small corpora are often used to model speech understanding domains and because they present n-gram frequency distributions quite different from those of large corpora.

The second corpus contains computer-written radiological reports and was collected during the A.Re.S. (Automatic Reporting by Speech) project (Angelini *et al.*, 1994a), currently developed at IRST. This corpus, which provides the best estimation conditions, corresponds to a real application dictation task for which speech recognition results will be given (see following section).

The third corpus is the English language texts collection issued by the Universities of Lancaster, Oslo and Bergen (LOB). The LOB corpus was designed for linguistics purposes and contains excerpts of articles and books covering several subjects and writing styles. Even if the LOB corpus is certainly not adequate for n-gram statistical language modeling purposes, as it contains over 1 million words for a vocabulary of about 50,000 words, it provides a benchmark for across-site LM testing.

Comparison experiments have been performed for all estimation methods with the interpolated bigram scheme defined in Section 2. Performance and rank position of each LM are reported in Table IV. LM ranking has been assessed by pairwise comparing the cross-entropies $\hat{H}_{LM}$ (see Section 2) of all the LMs. Given two LMs a and b, the employed statistical test (Johnson & Wichern, 1992) considers the N differences $d_t = \log Pr^a(w_t/w_{t-1}) - \log Pr^b(w_t/w_{t-1})$ over a test set $W = w_1 \dots w_N$. Assuming the differences, d_t, represent independent observations from an $N(\delta, \sigma_d^2)$ distribution, the variable

$$t = \frac{\bar{d} - \delta}{s_d/\sqrt{N}},$$

where $\bar{d}$ and s_d are respectively the mean and the standard deviation of the sample $d_1, \dots, d_N$, has a t-distribution with $N - 1$ degrees of freedom. By carrying out α-level tests (with $\alpha = 0.01$) of $H_0 : \delta = 0$ versus $H_1 : \delta \neq 0$, the ranking reported in Table IV resulted. In particular, only for the first corpus differences are not very reliable, which is probably due to the small size of the test set.

(TABLE IV ABOUT HERE)

Results show a considerable difference between the two naive methods, A1 and S1, and the other more refined methods. Interestingly, the Sβ method is the

LM that performs best, showing the highest robustness against data sparseness and frequency distribution noise. In fact, the GT-based LM, which performs very well on the large data corpora, falls slightly behind on the smallest corpus in which the frequency distribution d_i is probably not well approximated by countings. LE performs well on the small corpus but worsens a little as corpus size increases. This fact was also pointed out in Placeway *et al.* (1993). The LG Stacked model performs slightly worse than the Sβ one and shows good robustness. Finally, the LG Stacked model slightly surpasses the LG LOO one, which does not make use of a cross-validation set.

5.2 Network Compiling

To compare different LM topologies, the LG Stacked bigram LM trained on the 10,000-word A.Re.S. was mapped into three different networks: linear, tree-based and optimized. Words were phonetically transcribed by means of 50 context independent units. For each vocabulary entry, a little network for modeling breaths, pauses, hesitations was introduced.

The linear representation was built using the description in Section 4.1. The tree representation was built by applying the methods in Sections 4.2-4.4. The lexicon tree required 43,600 arcs. The average number of arcs in the successor trees was 73: so, the final network required only about 840,000 arcs against the almost $450 \cdot 10^6$ arcs needed by a full tree representation. The optimized network was built by applying the optimization algorithm described in Section 4.5.1 to the tree-based one. The network contained about 136,000 different symbols (*phoneme,probability*). The optimization process required 35 hours of CPU time on a HP-735 workstation. The optimization algorithm was implemented (Antoniol, Carli, Cettolo & Fiutem, 1992) to achieve a trade-off between time and space requirements.

The sizes of the three networks are reported in Table V. Labeled and empty arcs are counted separately, since they present a different computational cost during the speech decoding process.

(TABLE V ABOUT HERE)

The reduction provided by the optimization step is remarkable: the final number of labeled arcs is 2.6 times smaller than the first one.

The result confirms the importance of having explicitly represented the whole search space through a network in such a way that it can be reduced by a preprocess with acceptable time and space requirements.

The subtree isomorphism-based reduction gave only a slightly different result. The final size was 3% worse than that obtained by the optimization process. This means that the redundancy within the starting network is mainly due to the subtree isomorphisms. On the other hand, since time and space requirements of the two reduction algorithms are about the same, global optimization remains preferable.

5.3 Recognition Tests

5.3.1 System Description

Acoustic modeling uses a phonetic transcription of words with 50 context-independent units. Unit HMMs have simple left-to-right topologies of three or four states, depending on the average length of the corresponding units. Distributions are Gaussian mixtures with a variable number of components, resulting from a training process that initializes all mixtures with 24 components, and then prunes less used Gaussians. The final configuration used in the experiments reported includes a total of 2863 Gaussians grouped in 281 mixtures. The signal processing front-end provides the recognizer with a 27-dimensional vector every 10ms, consisting of 8 MEL scaled cepstral coefficients, the log-energy, and their first and second time derivatives. The acoustic parameter vector is scaled to ensure that all its elements have comparable ranges. In evaluating Gaussian mixtures, instead of summing all the terms, the approximation of taking the most likely term is made, since this allows a time gain without affecting accuracy.

Acoustic models were trained with maximum-likelihood estimation on a set of about 2000 sentences belonging to a phonetically rich database under collection at IRST (Angelini *et al.*, 1994b). The sentences and the speakers in the training set do not have any relation with the application domain.

The implementation of the decoding algorithm takes into account the fact that the network can be huge. Hence, in spite of the network being statically represented, the memory used in intermediate computations is allocated on demand with a simple caching strategy. Moreover, caching of distribution values is performed, ensuring that each distribution is computed at most once in every frame even if the same model appears on many arcs.

5.3.2 LM Comparison on Recognition Accuracy

All LMs were evaluated on the 10,000-word A.Re.S. task in terms of recognition accuracy. Tests were performed on a set of 759 radiological reports, amounting to 4 hours and 44 minutes of speech, recorded by 4 physicians (3 males and 1 female), only one of whom (*enmo*) had some experience in machine dictation. The reports were chosen with the aim of stressing acoustic variability, trying to include as many words as possible. This fact is reflected by the LM perplexities which are higher (see column PP in Table VI) than on the "linguistic" randomly chosen test set. Recognition results confirm the difference between the naive methods, A1 and S1, and the refined ones, which exhibit almost equivalent performance. The same statistical test of Section 5.1 was employed to assess differences of recognition performance between pairs of LMs. By considering sentence by sentence differences of word accuracy (WA), 0.05-level tests confirmed differences between A1 and the other LMs, between S1 and the other LMs, between GT and LS, and between LE and LS.

(TABLE VI ABOUT HERE)

5.3.3 Representation Comparison on Recognition Performance

Recognition tests were performed with the three different search space organizations: linear, tree-based, and optimized. Both acoustic framework and LM (LG Stacked) were kept fixed, and the beam threshold was chosen to achieve real-time response on an HP735 workstation. The results, reported in Table VII, refer to the networks presented in Section 5.2.

(TABLE VII ABOUT HERE)

Word accuracy obtained with the linear representation is worse than that of tree-based recognition, given the different impact of the beam threshold on the two topologies. Moreover, due to the higher average number of hypotheses per frame, the recognition is 5 times slower and the dynamic process size is larger even though the linear network is the smallest one. As can be seen from the table, the good properties of tree-based structure are preserved by the optimization procedure, and the memory requirements for network storage are drastically reduced. Recognition time, though considerably lower in the tree-based networks, does not decrease proportionally with the average number of active arcs per frame. This is because Gaussian evaluation is the most time consuming operation in the recognition system, and the linear network benefits from distribution values caching more than the tree-based networks do, since it generates a higher average ratio between number of active arcs and number of different active models.

(FIGURE 7 ABOUT HERE)

In Figure 7 the evolution of the number of active arcs during the decoding of a speech segment is shown. The upper window shows the speech waveform, with the labeling and segmentation resulting from the decoding process. The lines in the second window show how the number of active arcs in the network varies with time: the dashed line refers to the linear network, while the solid line refers to the tree-like network. The third window shows the same values, but in a logarithmic scale, to highlight differences in low-valued regions. This example reflects a qualitative behavior that was consistently observed in many cases. As expected, the most ambiguous regions turn out to be between-word transitions, both actual or potential. The peaks of ambiguity are much more severe when using the linear net. In fact, the big difference between the two representations is confined to these regions, since, as is evident from the log plot, in the word-ending regions the number of active arcs may be lower for the linear net. The within-word peaks correspond to potential word boundaries. For example on the word *persistenza* ("persistence"), whose beginning parts *per* and *persiste* are words by themselves (respectively, "for" and "persists"), peaks in the number of active arcs arise in correspondence to their end times, marked by arrows in the upper window of Figure 7.

6 Conclusions

This work focused on the problem of estimating and representing the LM, which constrains a Viterbi based beam-search HMM recognizer. In particular, several bigram estimation techniques were considered as well as FSN representations of them.

Concerning LM estimation, two basic bigram computation schemes were presented: backing-off and interpolation. These schemes differ mainly in the way they combine two basic ingredients: the discounted bigram frequency distribution and the zero-frequency probability. Several methods for estimating such components were described and tested within the interpolation scheme, both on written corpora, presenting varying data sparseness, and on a 10,000-word dictation task.

Experiments on corpora show that small but statistically significant differences appear among the best estimations, while the naive methods considered here ("add-1" and "shift-1") perform worse than these. A gap between these two groups of LM estimations was also measured in the dictation experiments. Among the best performing LMs is the one trained with the original Stacked estimation algorithm, proposed in this paper.

If a beam-search algorithm is used in the decoding process, the topology of the FSN representing the LM considerably influences the recognition time. The interpolation scheme is shown to be better suited to a Viterbi-based search as it provides a very efficient LM factorization. Two topologies have been evaluated:

linear and tree-based. The former implements the simplest FSN exploiting the above factorization, the latter is a refined version that adopts trees to represent the lexicon and all the explicited bigrams. Without any loss in recognition accuracy, the latter representation gives a 5-fold increase in recognition time. On the 10,000-word applicative task, this results in reaching real-time response on a HP-735, with 93% word accuracy.

If the search space is represented statically, FSN size becomes an important factor when the task complexity grows. An off-line reduction of the static tree-based network was presented as a viable method to overcome the problem of memory size. In the above application, a 60% reduction was achieved in the network size as well as in the dynamic size of the recognition process.

These results will be exploited for tasks with larger lexicons and perplexity, in which a two stage process of word hypotheses generation will be adopted.

References

A. Aho, J. Hopcroft, and J. Ullman, (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass.

B. Angelini, G. Antoniol, F. Brugnara, M. Cettolo, M. Federico, R. Fiutem, and G. Lazzari, (1994). Radiological reporting by speech recognition: the A.Re.S system. In *Proceedings of the International Conference of Spoken Language Processing*, 1267–1270, Yokohama, Japan.

B. Angelini, F. Brugnara, D. Falavigna, D. Giuliani, R. Gretter, and M. Omologo, (1994). Speaker independent continuous speech recognition using an acoustic-phonetic Italian corpus. In *Proceedings of the International Conference of Spoken Language Processing*, 1391–1394, Yokohama, Japan.

G. Antoniol, G. Carli, M. Cettolo, and R. Fiutem, (1992). Tools for development, test and analysis of ASRs. In *Proceedings of the Internation Conference on Signal Processing Applications and Technology*, 1005–1010, Cambridge, Mass.

L. E. Baum and J. A. Egon, (1967). An inequality with applications to statistical prediction for functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, **73**, 360–363.

L. E. Baum, (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov process. In *Inequal-*

ities 3, 1–8.

L. Breiman, (1992). Stacked regressions. Technical Report 367, Dept. of Statistics, University of California, Berkeley, CA.

P. F. Brown, S. F. Chen, S. A. Della Pietra, V. J. Della Pietra, A. S. Kehler, and R. L. Mercer, (1994). Automatic speech recognition in machine-aided translation. *Computer Speech and Language*, **8**, 177–187.

A. Corazza, M. Federico, R. Gretter, and G. Lazzari, (1993). Design and acquisition of a task-oriented spontaneous-speech data base. In V. Roberto, editor, *Intelligent Perceptual Systems*, volume 745 of *Lecture Notes in Artificial Intelligence*, 196–210. Springer Verlag, Heidelberg, Germany.

T. M. Cover and J. A. Thomas, (1991). *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons.

A.-M. Derouault and B. Merialdo, (1986). Natural language modeling for phoneme-to-text transcription. *IEEE Trans. Pattern Anal. Machine Intell.*, **PAMI-8**, 6 , 742–749.

M. Federico, (1993). Stacked estimation of interpolated n-gram language models. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition*, Snowbird, UT.

I. J. Good, (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, **40**, 237–264.

F. Jelinek and R. L. Mercer, (1980). Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice*, 381–397, Amsterdam, Holland.

F. Jelinek, R. L. Mercer, and S. Roukos, (1992). Principles of lexical language modeling for speech recognition. In S. Furui and M. M. Sondhi, editors, *Advances in Speech Signal Processing*, 651–699. M. Dekker, Inc., New York, NY.

R. A. Johnson and D. W. Wichern, editors, (1992). *Applied Multivariate Statistical Analysis.* Prentice Hall, Englewood Cliffs, NJ.

S. M. Katz, (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoust., Speech and Signal Proc.*, **ASSP-35**, 3 , 400–401.

K. Kukich, (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, **24**, 4 , 377–439.

P. Linz, (1990). *An Introduction to Formal Languages and Automata.* D.C. Heat and Company, Lexington, MA.

H. Murveit, P. Monaco, V. Digalakis, and J. Butzberger, (1994). Techniques to achieve an accurate real-time large-vocabulary speech recognition system. In *Proceedings of the ARPA Human Language Technology Workshop*, 368–373, Plainsboro, NJ.

A. Nadas, (1985). On Turing's formula formula for word probabilities. *IEEE Trans. Acoust., Speech and Signal Proc.*, **ASSP-32**, 1414–1416.

H. Ney, (1993). Architecture and search strategies for large-vocabulary continuous-speech recognition. In *New Advances and Trends in Speech Recognition and Coding*, NATO-ASI lectures. Springer Verlag, Berlin, Germany.

H. Ney, U. Essen, and R. Kneser, (1994). On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, **8**, 1–38.

H. Ney, R. Haeb-Umbach, B.-H. Tran, and M. Oerder, (1992). Improvements in beam search for 10000-word continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume I, 9–12, San Francisco, CA.

H. Ney, D. Mergel, A. Noll, and A. Paeseler, (1992). Data driven search organization for continuous speech recognition. *IEEE Transactions on Signal Proccessing*, **40**, 2 , 272–281.

P. Placeway, R. Schwartz, P. Fung, and L. Nguyen, (1993). The estimation of powerful language models from small and large corpora. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume II, 33–36, Minneapolis, MN.

V. Steinbiss, B.-H. Tran, and H. Ney, (1994). Improvements in beam-search. In *Proceedings of the International Conference of Spoken Language Processing*, 2143–2146, Yokohama, Japan.

V. Vapnik, (1982). *Estimation of Dependences Based on Empirical Data.* Springer Verlag, New York, NY.

I. H. Witten and T. C. Bell, (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Trans. Inform. Theory*, **IT-37**, 4 , 1085–1094.

A Bayesian Estimation

As both the Bayesian and the Good-Turing estimators are general techniques applicable to any discrete distribution, a more general estimation problem is considered here:

> given a population $V = \{v_1, \ldots, v_k\}$ and a sample $S = s_1, \ldots, s_n$ of n independent identically distributed (iid) random variables, find an estimate of the distribution $Pr(v)$ on V satisfying a suitable criterion.

The Bayesian estimation method (Vapnik, 1982) considers the following parametric statistics problem: find a distribution function $\pi(v \mid S)$ on V in a parametric family $Pr(v; p)$, with $p \in R^k$, by means of sample S, which minimizes the functional:

$$R_B(\pi) = \int [Pr(v \mid p) - \pi(v \mid S)]^2 Pr(S \mid p)\ Pr(p)\ dv\ dp\ ds_1 \ldots ds_n$$

where the *a priori* density $Pr(p)$ is supposed to be known. It can be shown that the minimum is attained for:

$$\pi^*(v \mid S) = \int Pr(v \mid p) Pr(p \mid S) dp$$

Note that the solution does not necessarily belong to $Pr(x; p)$.

By assuming that $p = (p(v_1), \ldots, p(v_k))$ is uniformly distributed on the simplex $C = \{p : \sum_{v \in V} p(v) = 1, p(v) \geq 0\}$, it results that

$$\pi^*(v \mid S) = \frac{c(v)+1}{n+k}$$

A proof of this results follows now (Vapnik, 1982).

From the assumptions on p it follows that $Pr(v \mid p) = p(v)$ and $Pr(p) = \frac{1}{vol(C)}$. By applying Bayes' rule

$$Pr(p \mid S) = \frac{Pr(S \mid p)Pr(p)}{\int_C Pr(S \mid p)Pr(p)dp}$$

the solution becomes:

$$\pi^*(v \mid S) = \frac{\int_C Pr(v \mid p)\ Pr(S \mid p)\ Pr(p)dp}{\int_C Pr(S \mid p)P(p)dp}$$

By computing the above integrals separately, it follows that:

$$\begin{aligned}
\int_C Pr(S \mid p)Pr(p)dp &= \int_C Pr(s_1, \ldots, s_n \mid p)Pr(p)dp \\
&= \frac{1}{vol(C)} \int_C \prod_{v \in V} p(v)^{c(v)} dp(v_1) \ldots dp(v_k) \\
&= \frac{1}{vol(C)} \frac{\Gamma(c(v_1)+1) \ldots \Gamma(c(v_k)+1)}{\Gamma(c(v_1)+\ldots+c(v_k)+k)}
\end{aligned}$$

and

$$\begin{aligned}
\int_C Pr(v \mid p)Pr(S \mid p)Pr(p)dp &= \int_C Pr(v \mid p)Pr(S \mid p)Pr(p)dp \\
&= \frac{1}{vol(C)} \int_C p(v) \prod_{v \in V} p(v)^{c(v)} dp(v_1) \ldots dp(v_k) \\
&= \frac{1}{vol(C)} \frac{\Gamma(c(v_1)+1) \ldots \Gamma(c(v_k)+1)(c(v)+1)}{\Gamma(c(v_1)+\ldots+c(v_k)+k+1)}
\end{aligned}$$

where $\Gamma(n+1) = n!$ if n is an integer. Hence, the Bayesian estimator easily follows:

$$\pi^*(v \mid S) = \frac{c(v)+1}{c(v_1)+\ldots+c(v_k)+k} = \frac{c(v)+1}{n+k}$$

B Good-Turing's Formula

This method was suggested to Good by Turing (Good, 1953) for the general problem of estimating the probabilities of species in a mixed population from sparse data. Given a population $V = \{v_1, \ldots, v_k\}$ and a sample $S = s_1, \ldots, s_n$ of iid random variables, the distribution $Pr(v)$ on V is estimated with the assumption that same frequencies correspond to equal probabilities (symmetry requirement).

The Good-Turing formula is:

$$\pi^*(v) = \frac{c^\star(v)}{n} = \frac{1}{n}(c(v)+1)\frac{d_{c(v)+1}}{d_{c(v)}}$$

where

$$d_r = \sum_{v \in V} \delta(c(v) = r)$$

is the number of elements of V occurring exactly r times in the sample S.

The proof of this formula as given by Good (1953) follows. Let $H = H(p(v_1), \ldots, p(v_k))$ be a statistical hypothesis asserting that $p(v_1), \ldots, p(v_k)$ are the population probabilities. The expected value of d_r in a sample of size n

is:

$$E_n(d_r \mid H) = \sum_{v \in V} E_n(c(v) = r \mid H) \quad = \quad \sum_{v \in V} \binom{n}{r} p(v)^r (1 - p(v))^{n-r}$$

The *a-posteriori* probability of a randomly chosen type $\hat{v}$ given that r occurrences were observed in a sample of size n is now considered. Let q_r be the probability value taken by a word occurring r times in S. By assuming that all p-s are different (also for very small values), and that $\hat{v}$ was chosen randomly, the desired *a-posteriori* probability can be expressed as follows:

$$Pr(q_r = p(\hat{v}) \mid r, H) = \frac{Pr(c(\hat{v}) = r \mid \hat{v}, H)\ 1/k}{\sum_{v \in V} Pr(c(v) = r \mid v, H)\ 1/k}$$

The likelihood of r given type $\hat{v}$ results:

$$Pr(c(\hat{v}) = r \mid H, \hat{v}) = \binom{n}{r} p(\hat{v})^r (1 - p(\hat{v}))^{n-r}$$

Hence the *a-posteriori* probability can be expressed as follows:

$$\begin{aligned} Pr(q_r = p(\hat{v}) \mid r, H) &= \frac{\binom{n}{r} p(\hat{v})^r (1 - p(\hat{v}))^{n-r}}{\sum_{v \in V} \binom{n}{r} p(v)^r (1 - p(v))^{n-r}} \\ &= \frac{p(\hat{v})^r (1 - p(\hat{v}))^{n-r}}{\sum_{v \in V} p(v)^r (1 - p(v))^{n-r}} \end{aligned}$$

Now, the expected value of q_r on a sample of size n is considered:

$$\begin{aligned} \hat{q}_r = E_n(q_r \mid r, H) &= \sum_{v \in V} Pr(v \mid H) Pr(q_r = p(v) \mid r, H) \\ &= \frac{\sum_{v \in V} p(v)^{r+1} (1 - p(v))^{n-r}}{\sum_{v \in V} p(v)^r (1 - p(v))^{n-r}} \end{aligned}$$

$$= \quad \frac{r+1}{n+1} \frac{\sum_{v \in V} \binom{n+1}{r+1} p(v)^{r+1} (1-p(v))^{(n+1)-(r+1)}}{\sum_{v \in V} \binom{n}{r} p(v)^{r} (1-p(v))^{n-r}}$$

$$= \quad \frac{r+1}{n+1} \frac{E_{n+1}(d_{r+1} \mid H)}{E_n(d_r \mid H)}$$

By assuming n large, the GT formula follows:

$$c^\star(r) \quad = \quad n * \hat{q}_r = \frac{n}{n+1} \; (r+1) \frac{E_{n+1}(d_{r+1} \mid H)}{E_n(d_r \mid H)}$$

$$\approx \quad (r+1) \frac{E_n(d_{r+1} \mid H)}{E_n(d_r \mid H)} \approx (r+1) \frac{d_{r+1}}{d_r}$$

C Proof of Property 1

Inequality 5 in Section 2 can be proved by application of the Jensen inequality (Cover & Thomas, 1991) to the function:

$$f(X = x_1, \ldots, x_n) = \left(\prod_{k=1}^{n} x_i \right)^{-\frac{1}{n}}$$

which is shown to be convex in the region $A = \{X : X \geq 0\}$. It is sufficient to show that $\log f(X)$ is convex as the exponential function preserves convexity. Convexity of $\log f(X) = \frac{1}{N} \sum_{i=1}^{N} - \log x_i$ easily follows from convexity of $-logx$ (see Cover & Thomas, 1991).

D LG Interpolation Leaving-One-Out Estimator

Training of LG interpolated LMs has been derived as a special case of the forward-backward training algorithm for HMM (Baum, 1972; Jelinek *et al.*, 1992) or from the LOO estimation framework (Ney *et al.*, 1994). A derivation is provided here that directly follows from a theorem by Baum & Egon (1967).

An LG interpolated n-gram LM can be defined as follows:

$$P_\lambda(z \mid s) = \sum_{i=1}^{m} \lambda_i(y) f_i(z \mid s)$$

satisfying the conditions:

$$\begin{aligned} \sum_{i=1}^{m} \lambda_i(y) &= 1, \quad \forall y \\ \lambda_i(y) &\geq 0, \quad \forall y, i = 1, \ldots, m \end{aligned}$$

where y is the Context Equivalence Class (CEC) of n-gram sz and f_i are appropriate (e.g. frequency based) approximations of $Pr(z \mid s)$ - e.g. trigrams, bigrams and unigrams conditional frequencies. A CEC class for a trigram xyz could be for example y or an appropriate discrete function on xy (Jelinek *et al.*, 1992).

Let the n-gram conditional frequencies be already computed over a training sequence W, the aim is to find values for the array of parameters $\lambda(y) = \{\lambda_i(y)\}$ $i = 1, \ldots, m$ which maximize the leaving-one-out likelihood function:

$$L(\lambda(y)) = P_\lambda(S_y) = \prod_{sz \in S_y} \left(\sum_{i=1}^{m} \lambda_i(y) f_i^*(z \mid s) \right)$$

where S_y denotes the set of all occurrences of n-grams sz with CEC y and f_i^* is the i-th approximation computed on W after deleting an occurrence of sz. For clarity, let y be considered implicit in the following.

It can be shown that the function L is a *homogeneous polynomial* of degree $|S_y|$ in its variables $\{\lambda_i\}$ and that the following theorem (Baum & Egon, 1967) can be applied.

Theorem. For any point $\lambda = \{\lambda_i\}$ in the domain D defined by the conditions (6), let $\tau(\lambda) = \tau(\{\lambda_i\})$ denote the point of D whose i-th coordinate is

$$\tau(\lambda)_i = \frac{\lambda_i \frac{\partial L}{\partial \lambda_i}}{\sum_{j=1}^{m} \lambda_j \frac{\partial L}{\partial \lambda_j}}.$$

Then $L(\tau(\lambda)) > L(\lambda)$ unless $\tau(\lambda) = \lambda$.

A proof of this theorem is given in (Baum & Egon, 1967).

It is worth noticing that the properties of map τ are maintained when $\log L$ is taken in place of L, as only a common factor $1/L(\lambda)$ results on both sides of the fraction. It follows that:

$$\lambda_i \frac{\partial \log L}{\partial \lambda_i} = \sum_{sz \in S_y} \frac{\lambda_i f_i^*(z \mid s)}{\sum_{j=1}^{m} \lambda_j f_j^*(z \mid s)}$$

and

$$\sum_{k=1}^{m} \lambda_i \frac{\partial \log L}{\partial \lambda_i} = |S_y|.$$

In this case:

$$\tau(\lambda)_i = \frac{1}{|S_y|} \sum_{sz \in S_y} \frac{\lambda_i f_i^*(z \mid s)}{\sum_{j=1}^{m} \lambda_j f_j^*(z \mid s)}$$

The right most expression is equivalent to the iterative estimator presented in Section 2. Besides, the factor $1/|S_y|$ can be omitted if a renormalization of the new $\tau(\lambda)_i$ is carried out after each iteration.

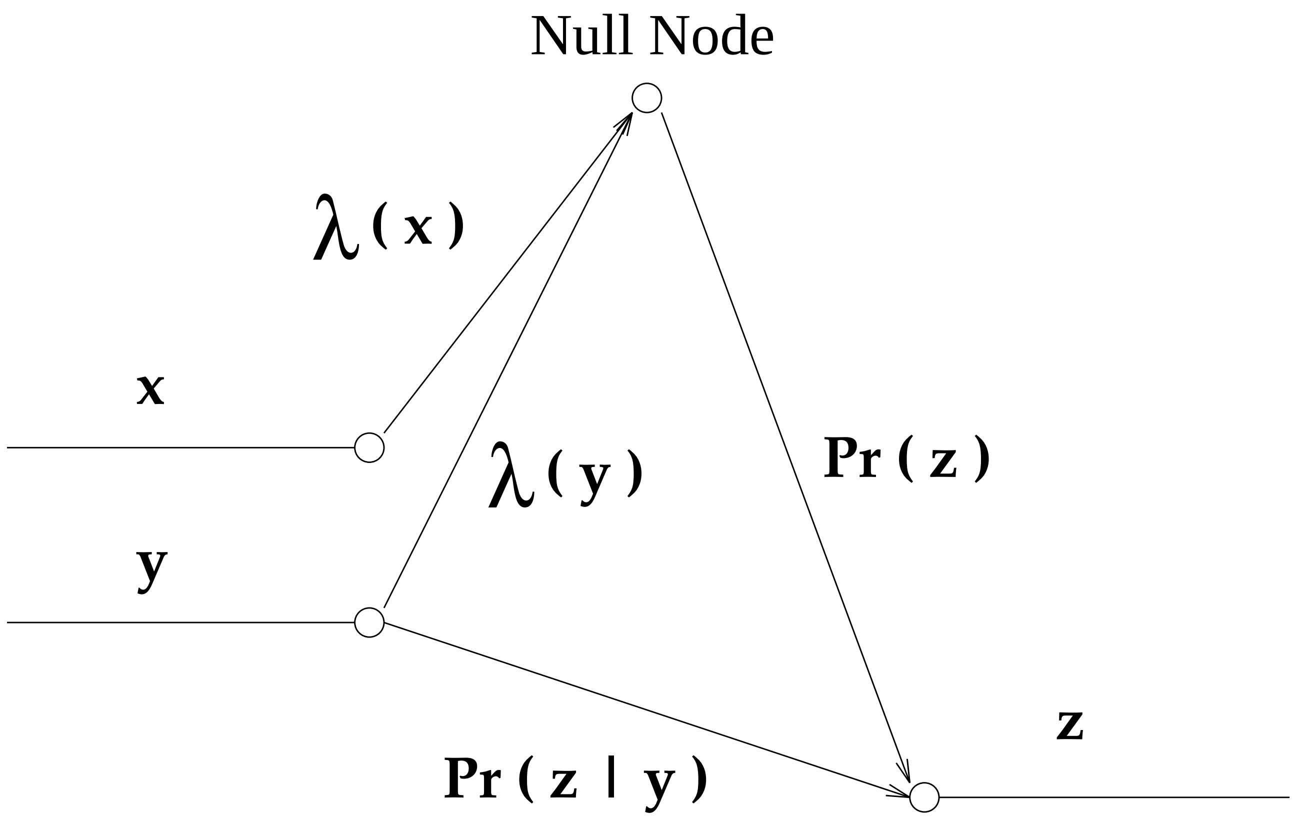

Figure 1: Bigram representation by using the "Null Node".

Table I: Notation.

V	vocabulary set
yz	bigram
$y_{\_}$	any bigram starting with y
$c(\cdot)$	number of occurrences in a text
$d(\cdot)$	number of different occurrences
$d_i(\cdot)$	number of different i-time occurrences
c, d_i	number of bigram occurrences and different i-time occurrences

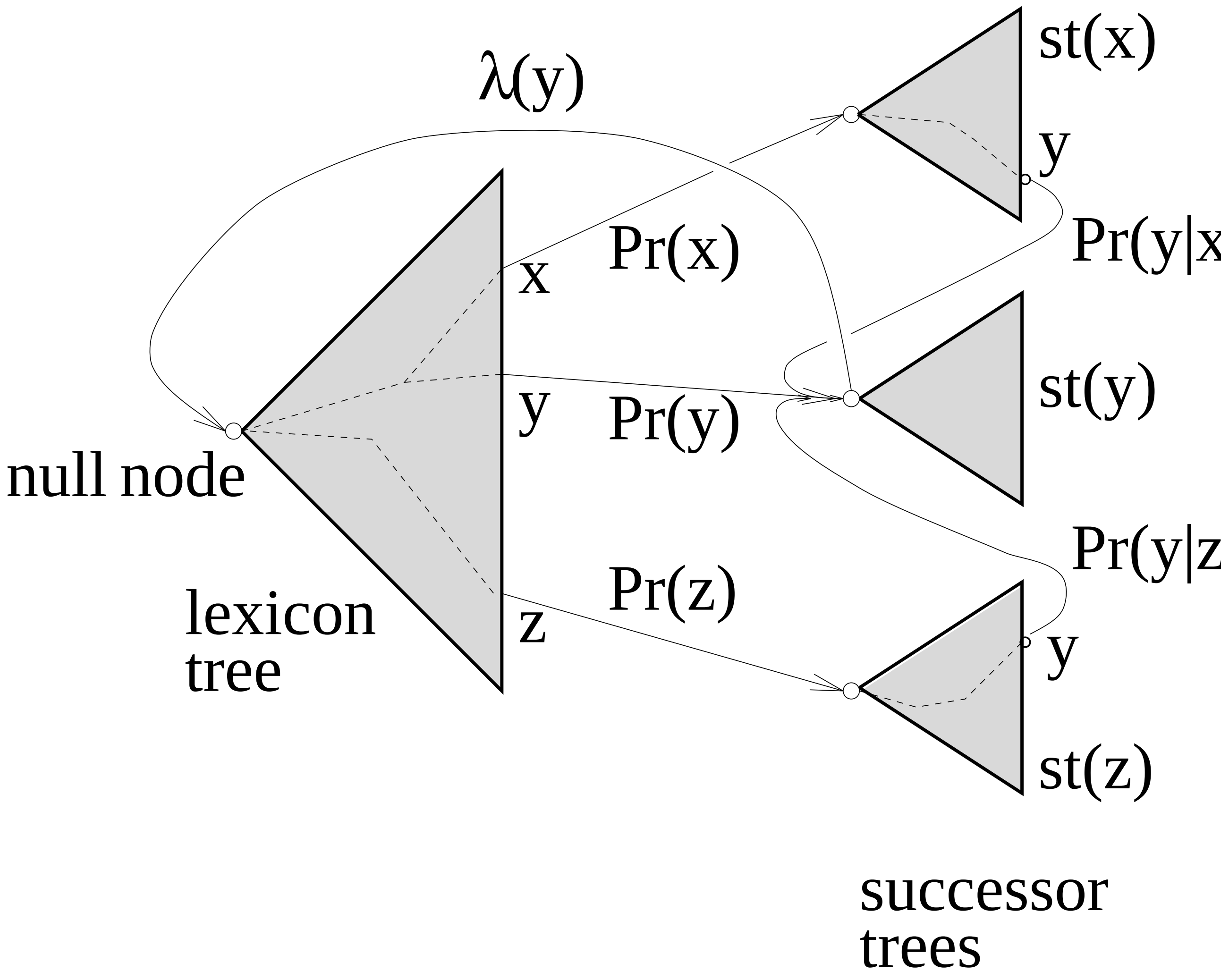

Figure 2: Tree-based Representation.

Initialization:

Let T indicate the starting tree, r its root, $S(a)$ the set of successor states of state a, and $p(a,b)$ the probability of the arc from a to b.

Algorithm:

```
**Factorize**(T)
    $\forall\ n\ \in S(r)$ UpDate(r,n)

**UpDate**($a$,$b$)
    if $b$ is leaf of $T$ then return
    $\forall\ s \in S(b)$ UpDate($b$,$s$)
    $\bar{p} \leftarrow \max_{s\in S(b)} p(b,s)$
    $\forall\ s \in S(b)$ $p(b,s) \leftarrow p(b,s)/\bar{p}$
    $p(a,b) \leftarrow \bar{p}$;
```

Figure 3: Algorithm for probability factorization in a tree.

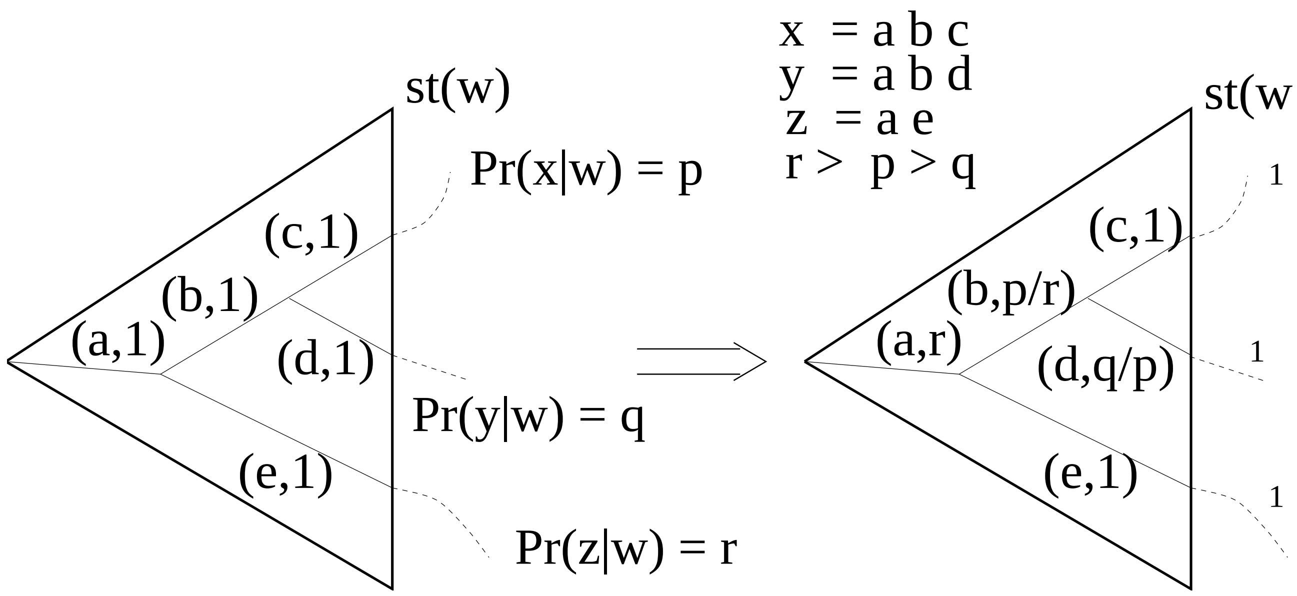

Figure 4: Effect of probability factorization.

Initialization:

Let a_h, $h = 0, \cdots, m-1$, indicate the m symbols labeling links.

Let $\Pi = \{B[i],\ i = 0, \cdots, n-1\}$ be an initial partition of states.

Let $W = \{(i,h) : i = 0, \cdots, n-1;\ h = 0, \cdots, m-1\}$ be the set of all active block-symbol pairs $B[i]$ and a_h.

Algorithm:

```
$\forall$ pair $(i,h) \in W$ until $W \neq \emptyset$
    Remove $(i,h)$ from $W$
    Let $I$ the set of states having an outgoing arc
    labeled with $a_h$ to some state of block $B[i]$
    $\forall$ block $B[j]$ such that $B[j] \cap I \neq \emptyset$ and $B[j] \not\subset I$
        Create a new block $B[n] \leftarrow B[j] \cap I$
        Update $B[j] \leftarrow B[j] - B[n]$
        $\forall$ symbol $a_k$
            if $(j,k) \in W$
            then add $(n,k)$ to $W$
            else if $\|B[j]\| \leq \|B[n]\|$
                then add $(j,k)$ to $W$
                else add $(n,k)$ to $W$
        $n \leftarrow n + 1$
```

Figure 5: Partitioning algorithm.

Initialization:

Let r indicate the root of the starting tree T, $T(v)$ its subtree starting from node v, and $I(v)$ the set of internal nodes of $T(v)$.

Algorithm:

```
$\forall s \in I(r)$
    $\forall v \in I(r) : v \neq s$
        if $T(v)$ and $T(s)$ are isomorph then
            Identify $v$ with $s$ in T and remove $T(v)$ from $T$
```

Figure 6: Sketch of the subtree isomorphism based algorithm.

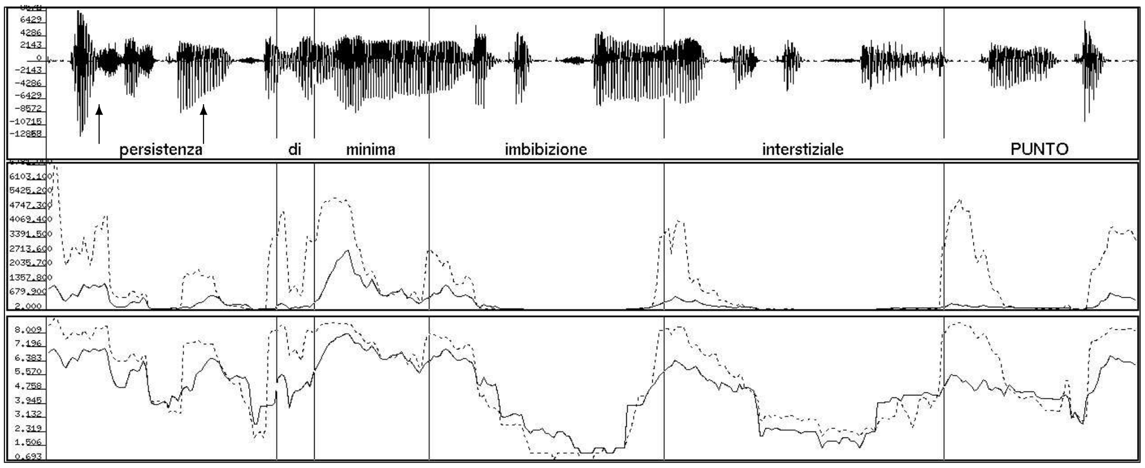

Figure 7: Number of active arcs during the decoding of a speech segment with two networks. See the text for a description.

Table II: Estimators for the discounted function $f^*(z \mid y)$ and the zero frequency probability $\lambda(y)$.

Discounting Model	$f^*(z \mid y)$	$\lambda(y)$	Remarks
A1	$\delta(c(yz))\frac{c(yz)+1}{c(y)+\lvert V\rvert}$	$\frac{d_0(y_-)}{c(y)+\lvert V\rvert}$	
GT	$h(yz)f(z \mid y)$	$1-\sum_z f^*(z \mid y)$	$h(yz)=\frac{d_{c(yz)+1}}{d_{c(yz)}}\frac{c(yz)+1}{c(yz)}$
S1	$\max\left\{\frac{c(yz)-1}{c(y)},0\right\}$	$\frac{d(y_-)}{c(y)}$	
Sβ	$\max\left\{\frac{c(yz)-\beta}{c(y)},0\right\}$	$\beta\frac{d(y_-)}{c(y)}$	$\beta \approx \frac{d_1}{d_1+2d_2} < 1.$
LE	$\frac{c(y,z)}{c(y)+d(y_-)}$	$\frac{d(y_-)}{c(y)+d(y_-)}$	
LS	$(1-\alpha)f(z \mid y)$	α	$\alpha = \frac{d_1}{c}.$
LG	$(1-\lambda(y))f(z \mid y)$	$\lambda(y)$	

Table III: Statistics of the corpora used.

Corpora features	Concierge	Radiology	English
Content	queries	reports	generic texts
Vocabulary size	907	10,261	49,615
Corpus size	26,455	1,893,286	1,157,260
Training text size	18,030	1,417,285	866,980
Testing text size	8,425	476,001	290,280
Bigram coverage-rate	85%	94%	71%

Table IV: Perplexity measures and rankings of LMs with different texts.

Model	Concierge	A.Re.S.	LOB
A1	57.58_7	45.47_8	791_8
GT	20.83_3	17.52_2	422_2
S1	23.05_6	18.55_7	472_7
Sβ	20.47_1	17.50_1	421_1
LE	20.53_1	17.57_3	443_5
LS	21.63_5	18.14_6	465_6
LG LOO	21.46_5	17.63_5	441_4
LG Stacked	20.77_3	17.59_4	431_3

Table V: Linear, tree-based and optimized network sizes.

Topology	#States	#Labeled arcs	#Empty arcs	Total arcs
Linear	134,439	124,002	165,478	289,480
Tree-based	687,329	821,626	20,744	842,370
Optimized	195,012	311,129	15,659	326,788

Table VI: Speaker and average LM recognition performance in terms of WA. Perplexity of the uttered sentences is also provided.

	clfu	enmo	euma	lube	avg.	PP
A1	81.48	90.28	90.13	84.91	86.70	61.13
GT	89.82	94.73	94.48	92.16	92.80	26.62
S1	88.91	94.13	93.87	91.15	92.01	28.35
Sβ	89.85	94.65	94.59	92.02	92.78	26.50
LE	89.97	94.67	94.98	92.66	93.07	26.32
LS	89.48	94.77	93.97	92.06	92.57	27.82
LGL	89.68	94.68	94.59	92.75	92.93	26.46
LGS	89.70	94.70	94.88	92.62	92.97	26.33

Table VII: Comparison among linear, tree-based and reduced tree-based representations in terms of Word Accuracy (WA), real-time ratio and recognition process size.

Topology	# Arcs per Frame	WA	RealTime Ratio	Process Size (Mb)
Linear	2350	90.80	4.93	70
Tree-Based	292	92.97	1.01	58
Optimized	285	92.97	1.01	23