DYNAMIC LANGUAGE MODELS FOR INTERACTIVE SPEECH APPLICATIONS *

Fabio Brugnara and Marcello Federico

IRST - Istituto per la Ricerca Scientifica e Tecnologica
I-38050 Povo, Trento, Italy.
{brugnara,federico}@irst.itc.it

ABSTRACT

This work proposes the use of hierarchical LMs as an effective method both for efficiently dealing with context-dependent LMs in a dialogue system and for increasing the robustness of LM estimation and adaptation. Starting from basic LMs that express elementary semantic units, concepts, or data-types, sentence level LMs are recursively built. The resulting LMs may be a combination of grammars, word classes, and statistical LMs. Moreover, these LMs can be efficiently compiled into probabilistic recursive transition networks. A speech decoding algorithm directly exploits the recursive representation and produces the most probable parse tree matching the speech signal. The proposed approach has been implemented for a data-entry task which covers structured data, e.g. numbers, dates, and proper names, as well as free texts. In this task, the active LM must continuously change according to the current status, the active form, and the data entered so far. Finally, while the hierarchical approach results very convenient to cope with this task, it also looks very general and can give advantages in other applications, e.g. dictation.

1. INTRODUCTION

This work presents a technique for efficiently managing dynamic Language Models (LMs) in dialog or data-entry applications. When speech is used to input structured data the speech recognizer has to apply different LMs depending on the semantic type of the entered data. For instance, dates, numbers, codes, can be modeled with specific grammars, while texts are usually modeled with n-gram LMs. Moreover, in data-entry applications, the LM to be applied may not only depend on the data-type to be dictated but also on some context information and previously entered data. Especially when large vocabularies are involved, this approach requires the ability of efficiently combining basic LMs, which can be shared in different contexts, to instantiate context specific LMs. This naturally leads to a recursive representation of LMs.

This paper is organized as follows. Section 2 introduces hierarchical LMs and briefly discusses their application to n-gram models and to dynamic language modeling. Section 3 outlines the speech decoder used to search recursive networks. Section 4 describes the application of the above concepts within the SpeeData project, that deals with a complex data-entry task. Finally, Section 5 gives some conclusions.

*This work is supported by the European Commission, Telematics Application Programme, project reference number LE 1999.

2. HIERARCHICAL LMs

Hierarchical representations of language have been largely used in ASR and more specifically in spoken language understanding systems. In general, meaning units, i.e. concepts, are hierarchically defined from words, syntactic/semantic word classes, or even other concepts. Search for the best interpretation is usually performed on some word-based intermediate structure, e.g. a lattice or an n-best list, produced by a speech decoder exploiting some conventional LM. A *Probabilistic Finite State Network* (PFSN) with word-labeled arcs is normally employed which represents all possible ways of concatenating words [2]. Performing the interpretation stage directly on the speech signal requires the parsing process to be similar to a standard HMM decoding algorithm, and the LM representation to preserve the hierarchical nature. Both conditions can be met by using a generalization of PFSN, the so called *Probabilistic Recursive Transition Networks* (PRTNs), and by extending the Viterbi algorithm accordingly. The generalization of PFSN to PRTNs is obtained by allowing arc labels to denote other PFSNs. The extended Viterbi algorithm is briefly described in the next section.

In this paper, the application of hierarchical LMs is considered in two different contexts. First, a class of more powerful n-gram based LMs is introduced that is even suited for large vocabulary dictation and second, the application of hierarchical LMs in interactive spoken language applications is discussed.

2.1. Generalized class-based n-grams

An interesting extension to the ordinary class-based n-gram LMs is provided by allowing classes to represent either single words, lists of words, grammars, or other n-gram models. Of course, nothing prevents from having several levels of nested n-gram models. Clearly, such a model can be conveniently represented with a PRTN. This allows hierarchical LMs to be implemented without explicitly duplicating each class representation for every different context in which it occurs.

The LM probability of a word sequence W can be computed by taking into account all possible interpretation (parse) trees $T(W)$ of W by means of the LM, i.e.:

$$\Pr(W) = \sum_{T(W)} \Pr(T(W))$$

However, when using a hierarchical LM more information is contained in the most probable interpretation, than in the most probable word sequence. The extended Viterbi algorithm will thus compute:

$$T^* = \arg\max_T \Pr(T, A)$$

where A is the acoustic observation.

The probability of an interpretation tree can be computed in a recursive way. $T(W)$ can be expressed as a parse tree t_0 whose root c_0 has m sub-trees $t_1, \ldots, t_m$, with roots $c_1, \ldots, c_m$. The underlying idea is to see internal nodes of the tree as generalized classes and leaves as words. Hence, the probability of $T(W)$ is:

$$\Pr(T(W)) = \Pr(T = t_0 \mid c_0) \quad = \prod_{i=1}^{m} \Pr(t_i \mid c_i) \Pr(c_i \mid c_{i-n+1}, \ldots, c_{i-1})$$

$\Pr(t_i \mid c_i)$ is computed differently according to the nature of the class c_i. If c_i is modelled with a word list then t_i has only one leaf w and $\Pr(t_i \mid c_i) = \Pr(w \mid c_i)$. If c_i is modelled with a grammar, then t_i is a parse tree and $\Pr(t_i \mid c_i)$ is computed by following the grammar rules. If c_i is modelled with an n-gram model, then $\Pr(t_i \mid c_i)$ can be computed by recursively applying the above scheme to the list of successors of c_i.

An example will better clarify the above concepts. Let one consider the following sentence taken from the *Wall Street Journal* corpus:

THE STEEPEST FALL WAS AT BANKAMERICA CORP.'S BANK OF AMERICA A THIRTY PERCENT DECLINE TO TWENTY EIGHT MILLION DOLLARS FROM FOURTY MILLION DOLLARS

A possible interpretation by means of a generalized class-based LM could be the following one:

THE STEEPEST FALL WAS AT **COMPANY(** *BANKAMERICA CORP.'S BANK OF AMERICA) A* **PERC(** **NUM2(** *THIRTHY* **)** *PERCENT* **)** *DECLINE TO* **AMOUNT(** **NUM9(** *TWENTY EIGHT MILLION* **)** *DOLLARS*) *FROM* **AMOUNT(** **NUM9(** *FOURTY MILLION* **)** *DOLLARS* **)**

In the above interpretation, words in italics correspond to single word classes. The class **COMPANY** could be modelled with an ordinary n-gram model, while the **PERC** and **AMOUNT** classes are modelled with specific grammars. Both grammars refer to other grammars, i.e. **NUM9** and **NUM2**, which indicate numbers of at most 2 and 9 digits, respectively.

Generalized class-based LMs improve ordinary n-gram word or class based models in several ways.

With respect to data sparseness, the hierarchical LM is more robust as many content words or phrases, e.g. proper names, are mapped into specific classes. In fact, proper names typically have sparse frequency distributions, and only few contexts are observed.

If PRTNs are employed less space requirements are needed to represent these LMs. In fact, by introducing generalized classes, less n-grams need to be stored and each class representation is stored once.

The hierarchical LM is better structured and allows easier adaptation of probabilities and insertion of new terms. For instance, new names of companies can be added inside the corresponding class without affecting the highest LM statistics. Moreover, adaptation can be independently applied to the higher order LM and to each constituent classes. In this way, as fewer parameters are present in the higher order LM, a smaller amount of adaptation data is required. Besides, there are conditions in which it is natural to adapt a LM just by replacing some class LMs. For instance, in the SpeeData system the geographical names strongly depend on the district in which the data-entry is carried out, while the higher LM statistics are much less site-dependent.

Apart from the preparation of the single grammars, training of hierarchical LM can be carried out automatically. The requirement is to have a tagged corpus for each of the employed classes. Tagged corpora can be obtained, for instance, by means of pattern matching techniques. Moreover, class LMs can be automatically estimated from external sources, e.g. corpora or specialized thesauri. Single LMs can then be trained separately on each corpus by means of standard n-gram estimation algorithms, and compiled into specific PRTNs.

2.2. Dynamic LMs

Another application field of hierarchical LMs are dialogue based ASR systems. Phone services, information query systems, data-entry systems all require dynamic LMs that automatically vary according to the dialogue state. In general, the linguistic domain of such systems can be conveniently structured into a set of domain specific concepts. For instance, a stock market query system usually deals with concepts like names of stocks, names of stock exchanges, percentages, etc. Each concept can be easily represented with specific LMs. During the dialogue, dynamic LMs have to be built to model the expected input language. Hierarchical LMs allow the active LM to be expressed by just using references to the concept LMs. This greatly reduces the size of the resulting network so that it can be efficiently compiled on-the-fly. Section 4 will describe in detail an application of this technique.

3. RECURSIVE SPEECH DECODER

An effective use of the PRTN representation requires a speech decoder that directly exploits the recursive structure without needing its explicit expansion into an HMM based PFSN. This can be achieved by extending the standard Viterbi algorithm so as to allow recursion in the ex-

pansion of arcs during the network search procedure. A good description of the frame-synchronous Viterbi search applied to HMM-based networks can be found in [3]. The computation of the quantities needed to find the optimal path consists in performing, for every input frame, a dynamic programming procedure that expands all active hypotheses, i.e. it extends all the paths that lead to a state with a non-zero probability. This procedure is applied at two levels, the network-level, in which the probabilities of network states are considered, and the HMM-level, in which the internal states of the models are updated. The expansion of a network arc at time t requires to perform a step of the Viterbi algorithm inside the corresponding model. This internal step considers the likelihood of the origin network state, multiplied by the arc probability, as a possible score for a path entering the model at time t. The likelihood of the final state of the model at time $t+1$ is then taken as one of the possible scores for a path entering the destination network state at time $t+1$.

If the two levels are kept distinct, the upper level search algorithm, working on the network, does not depend on the internal structure of the objects associated to the arcs. It is only relevant that the internal search algorithm is able to correctly exploit the hypotheses of incoming paths and to provide an output that consists in an "optimal" accumulated likelihood. Hence, nothing prevents arcs from referring to other networks, instead of HMMs. This is precisely the concept behind the proposed PRTN decoder. In this algorithm, the expansion of a network arc can be performed either by a call to an HMM-specific function, or by a recursive call to the network-level expansion function itself, with the same input and output likelihoods. An analogous extension is needed for the backtracking procedure, which retrieves the most probable path after all frames have been processed. The recursive structure of the backtracking algorithm allows to build a parse tree of the input signal, since it makes possible to keep track of the network spanning each speech segment.

4. APPLICATION TO DATA-ENTRY

4.1. SpeeData Domain

This work is carried out within the SpeeData project [1], whose objective is the development of a demonstrator for multi-lingual spoken data-entry. The application domain is the electronic storage of the Land Register of an Italian bilingual region, where Italian and German are both official languages. The Land Register contains information about real estates for a total of about 600,000 documents. The database fields show a variety of formats:

- numbers: dates, portions, amounts of money, etc.;
- fixed texts: limited lists of expressions;
- proper names: owners, locations, etc.;
- free texts: descriptions of properties, rights, etc.

Dictionaries of the fields may vary from a few tens to a few thousands of words.

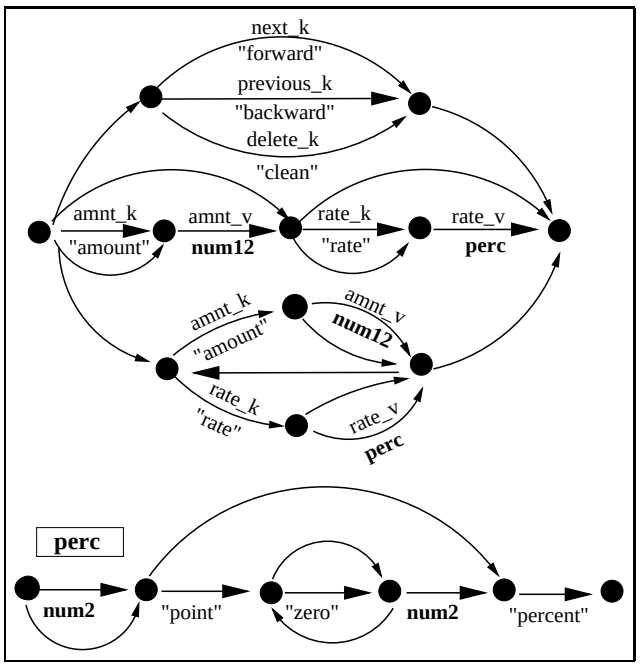

Figure 1: Architecture of the `SpeeData` demonstrator.

The data-entry process involves filling in several forms which can be brought up on the screen by uttering keywords. Each form contains different data-fields. A form can be filled in field-by-field, by uttering a field keyword followed by the field contents. More keywords and data can be dictated in continuous speech, so that several fields can be filled in with a single utterance. Alternatively, the user can explicitly select a specific field by isolately uttering the corresponding keyword. This modality is useful for correcting previous recognition errors since, in this case, a field-specific LM is activated. Each data-entry operations may have an effect on the active LM. In fact, the active LM is continuously updated on the basis of the current form, active field, and previously filled-in fields.

The LM is dynamically modified by generating a new syntactic expression that combines data-type and keyword specific grammars according to rules that take into account the current state of the interaction. The resulting regular expression is compiled at run-time into a PRTN.

In Figure 1, an example of a PRTN corresponding to an active LM is shown. For the sake of simplicity, a form is considered that contains only two fields, namely *amnt* (short name of *amount*) and *rate*, and three editing commands, namely *next*, *previous* and *clean*, that allow to move to other forms or to clean the current form or a selected field. Moreover, probabilities are omitted for reason of space. It can be noticed that each arc of the network may carry two labels: one denoting a PRTN (in boldface) or a word, and one denoting a semantic class. During the decoding stage, the semantic classes encountered in the best matching path of the LM are collected and used to qualify the output strings. The PRTN in the figure allows three main paths: a single command keyword, an ordered sequence of keyword-optional field assignments, an order-free sequence of keyword-mandatory field assignments or selections.

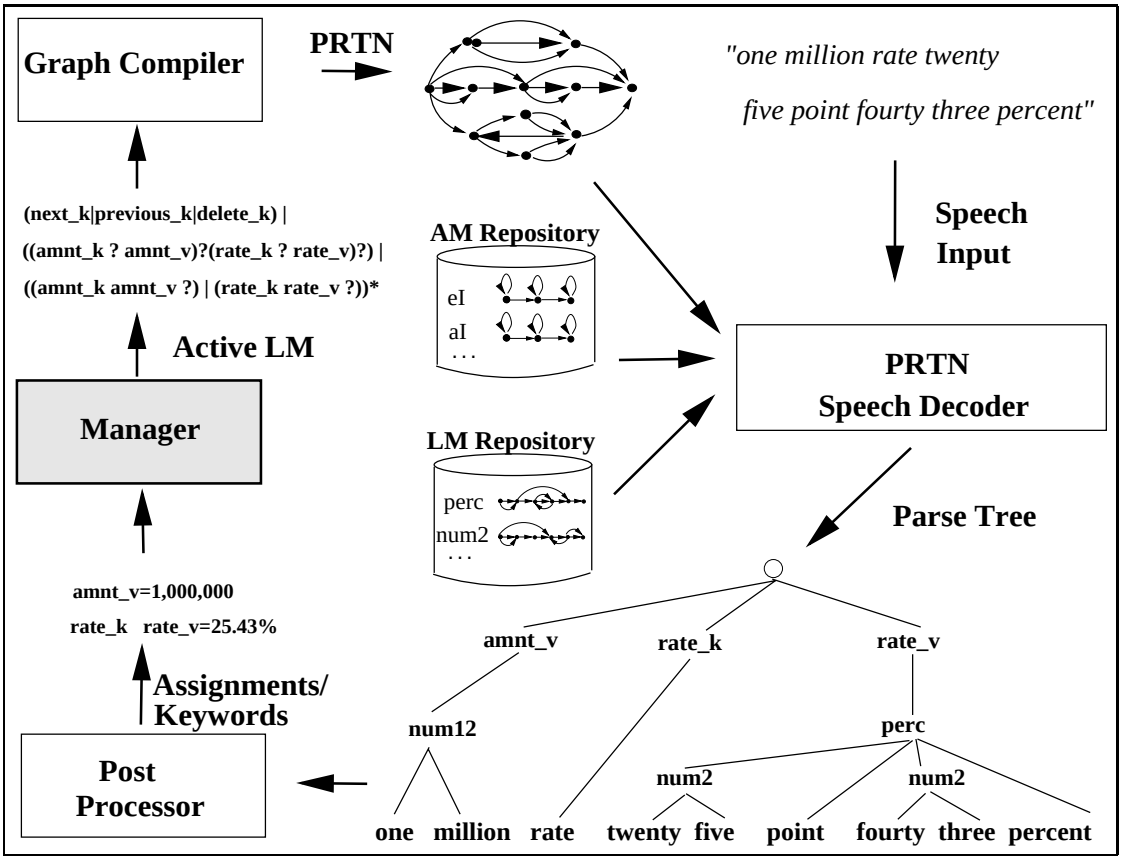

Figure 2: Data flow of the SpeeData application.

This LM would be different if a field was selected or filled-in. For instance, if the *rate* field was selected, the active LM would allow either to fill-in that field or to utter any isolated keyword.

In the SpeeData domain there are about 40 LMs corresponding to different data-types. These LMs are either hand-designed grammars, word lists, or statistically trained LMs. Each data-type LM is compiled off-line into a corresponding PRTN.

In Figure 1, the PRTN used to model the *rate* field is also shown. In fact, the **perc** (short name of percentage) PRTN itself makes use of another PRTN, namely *num*2, which models numbers between 1 and 99. Clearly, a high variety of LMs can be generated and efficiently represented with a limited number of pre-compiled PFSNs. In general, the definition of data-type LMs, e.g. texts and dates, can exploit other data-type LMs, e.g. numbers.

In Figure 2 a portion of the SpeeData architecture is shown. The diagram shows the components involved in the generation of the LM, in the speech recognition process, and in the post processing of the interpretation parse tree. The Manager module produces a state dependent LM, here called *active LM*, in terms of a regular expression. This syntactical expression is compiled at run-time into a PRTN and passed to the speech decoder. This network can refer to other PRTNs which have been preloaded by the Speech Decoder from a LM repository. The incoming speech utterance is decoded into a parse tree. This data structure is then explored by a Post Processor that transforms it into a sequence of data assignments and/or keywords. Finally, this list is returned to the Manager that performs the corresponding actions and varies the dialogue state accordingly. A description of the complete system can be found in [1].

5. CONCLUSION

This work has presented the application of hierarchical LMs and PRTNs to ASR. A practical use of this techniques to a speech based data-entry system has been described. The possible application of the same formalism to large vocabulary dictation has also been outlined. Work is under way for evaluating generalized class-based n-gram LMs on an Italian newspaper dictation task.

6. REFERENCES

[1] U. Ackermann, B. Angelini, F. Brugnara, M. Federico, D. Giuliani, R. Gretter, and H. Niemann. SpeeData: a prototype for multilingual spoken data-entry. In *Proc. EUROSPEECH*, Rhodes, Greece, 1997.

[2] M. Federico, M. Cettolo, F. Brugnara, and G. Antoniol. Language modeling for efficient beam-search. *Computer Speech and Language*, 9:353–379, 1995.

[3] C. H. Lee and L. R. Rabiner. A frame-synchronous network search algorithm for connected word recognition. *IEEE Trans. Acoust., Speech and Signal Proc.*, ASSP-37(11):1649–1658, 1989.