A Multi-level Lexical-semantics Based Language Model Design For Guided Integrated Continuous Speech Recognition

Francisco J. Valverde-Albacete,***

*José M. Pardo***

*Area de Tecnología Electrónica, Dept. Ingeniería. Univ. Carlos III, Madrid, Spain

** Grupo de Tecnología del Habla, Dept. Ingeniería Electrónica, Univ. Politécnica de Madrid, Spain

ABSTRACT

We present a continuous speech recognition architecture with a tightly coupled language model that tries to improve the dwindling performance of the normal stack decoder with increasing lexicon size. We solve the problem of recognition by means of two mutually recursive functions. The first one uses an auxiliary retrieval function to obtain lexicalized (already built) solutions to the problem, and merges these solutions with the ones built by the second function. This second one describes the acoustical and semantic recognition process as a search problem defined with the help of the first function, and solved with the help of the A* strategy. As a linguistic model, we use a hierarchy of linguistic levels each of which has a particular meaning structure, a lexicon of lexicalized forms, their lexicalization probabilities, and a local lexical grammar describing how the semantic categories of the level can be built. The process can further be optimized if *targets*, constraints on the possible solutions, are given to the recognition process to guide and restrict it. Target guidance implies a mechanism for *target focusing*, locally matching targets to the recognition state, and *target prediction* with the help of a lexical local grammar. We are testing the architecture in a DARPA RM-like application.

1. INTEGRATED CONTINUOUS SPEECH RECOGNITION WITH AN A* STACK DECODER

Although the Stack Decoder is a well-known algorithm for speech recognition [1.], not until recently has it been considered as interesting for being integrated with a language model. On top of being used as the second pass in a number of multipass, decoupled schemes, it has been proposed as a stand-alone algorithm for one-pass almost time-synchronous speech recognition([3.],[4.], [5.]).

1.1. The Basic A* Stack Decoder

D.B. Paul proposed as early as 1992 a stack decoder tightly-coupled or integrated architecture that used the A* strategy for implementing the search [5.]. A* is a graph search algorithm that adapts easily to an abstract specification of the search problem. It uses a scoring function *f* for deciding which node is to be expanded next: the one with the best (according to some criterion) *stack score* is then expanded. Function *f* accounts for two different scores about each node: the cost of reaching a node from the start of the search, and the estimated cost to reach a solution whatsoever from that node onwards. Under certain constraints on the estimation function the search if guaranteed to be complete and admissible [2.]. Paul also proposed several optimizations for the estimation function and the algorithm [4.].

More interesting is his concept of a tightly coupled architecture around the stack decoder operation in which an acoustical decoder and a linguistic model could cooperate to achieve better recognition, performance [5.]. Unfortunately, only unigrams and bigrams are reported as models for such a system ([3.]) which lessen the linguistic quality of the output, compared to a more sophisticated, speech parsing or understanding system.

2. THE STRUCTURE OF THE SYSTEM

The original stack architecture is what would be called nowadays an acoustically-driven tightly coupled system; tightly coupled because both acoustic and linguistic scores are considered at the same time when calculating the overall score of each hypothesis; acoustically driven because it is the acoustic front end who takes the responsibility of pruning and proposing alternatives leaving the linguistic model with only the scoring of them.

We are proposing an alternative, semantically driven architecture in which linguistic information can more easily and helpfully be introduced. The basic search algorithm will still be *A*.

2.1. Linguistic Signs

Our basics building units are full linguistic signs composed of a *significant* part (an acoustic image) and a *meaning* (a lexical meaning in fact). Furthermore, we will distinguish two different phases of behaviour in a sign: the *paradigmatic phase*, when it is used in opposition to other similar signs, and the *sintagmatic phase* when it is combined with other, similar signs to build new signs. More about that later.

2.2. Linguistic Levels

For the time being, let each sign belong to one of several disjoint sets; in the following, we will be considering sets of sentences, clauses, phrases and words (even though the scheme extends easily to morphemes, we will not be considering them).

Although no unit can belong to more than one set, adequate (grammatical) sequences of signs from one set can be *injected* into

another: i.e. [[the] [horse]] is a valid phrase built out of two words ([the] and [horse]).

Level lexicon. However, the (meaning) properties of some units cannot be explained in terms of the apparent sequences of constituents (such as [fishy affair]) so we must allow for the existence of a subset that includes all of this non-analyzable signs in each of the sentence, clause, phrase and word sets (thus introducing a flavour of non-compositional semantics). We will call each of this subsets the *lexicon* of the set.

Uppermost level. One of these sets is distinguished in that the results of recognition can only belong to it: let's say sentences. With this set on top we build a hierarchy of *levels* [8.] in which the units of one set are either contained in the lexicon, or built with the help of the one immediately below it:

- e.g: *sentence* > *clause* > *phrase* > *word*

Bottom level. The bottom of the hierarchy is another distinguished level in the sense that no new element can be created combining elements from an even lower level because none exists, so it amounts only to its lexicon. In the present hierarchy such is the case of the *word* level (but in a hierarchy including a morpheme level, that would be the bottom).

2.3. Acoustic Modelling

We model acoustical input as a (possibly erroneous) string or sequence of *phonemes*, and their deletion, insertion and substitution probabilities. Significants will be modelled by *prototypes*, non-void canonical strings of phonemes, and we have a DTW procedure for calculating the distribution of the distance of the alignment paths between a prototype and a phonetic sequence indexed on the time of the sequence.

Although we could have used a more complicate, closer to the state-of-the-art model for acoustic recognition (an HMM on acoustic frames, for instance) we have decided to concentrate on the linguistic model. We trust that our model of the acoustic input as a sequence of observations against which to match prototypes will prove general enough.

2.4. Semantic Modelling

Models for the semantic component of a sign are less known than their phonetic counterparts. We will introduce one of the many which have been proposed.

Paradigmatic meaning: sign senses. One of the models considers only the lexical meaning of the sign, that which is manifest when opposing any two signs belonging to a set. This is also called the *paradigmatic meaning* of a sign and is most often referred to as its *lexical meaning* ([6.] chaps. 1,3).

Among the several relationships which try to capture the nature of the opposition between senses we have decided to model only the hiperonimy-hiponimy-cohiponimy, and some of the opposition relationship ([6.] chap. 5-6, 9-10), which can be modelled to advantage by means of tree structures ([7.]).

Sintagmatical meaning: selectional constraints. We must furthermore accept another, *sintagmatical meaning*, which manifests itself when trying to build larger signs. This meaning is also a lexical meaning, but is usually referred to as the *selectional constraints* imposed on and by the sign ([6.] chaps. 1-2).

One way to record the collective selectional constraints of a set of signs is to model them with a grammar, such as a traditional Chomsky-type one, but this has the disadvantage of hiding the behaviour of the individual component signs. Instead, we will be using a type of *lexical grammar*, in which each sign denotes its own selectional restrictions.

3. THE RETRIEVE-AND-DELVE ALGORITHM

We will use a function for each purpose, as each function will only use either the paradigmatic behaviour or the sintagmatical behaviour of the signs we call them two different *phases* of recognition.

3.1. The Paradigmatic Phase: Sign Retrieval

We state the problem of speech recognition as that of finding the solution to an equation into which the sequence of acoustic observations and a starting level are fed: the solutions, or *hypothesis*, take the form of a triple *(p, x, u)* where *p*, denotes a score (probability, likelihood) for each solution, *x* denotes the final exploration state reached by the hypothesis and *u* is the sign constructed by the recognition process; internally, the hypothesis take the form of speech parse trees. In admitting several scored hypothesis as solutions we are adopting the N-best recognition model.

Starting point. The signs built during recognition must belong to the upper level of the hierarchy, that is to say, all signs that belong to the sentence level and "match" the observation input sequence must be found. However, for generality's sake, let us suppose that the recognition process starts in a level whatsoever and we have as input a local starting exploration state.

Early termination, recursion and merging. If the level is the bottom one then all solutions have to be lexicalized and contained in it, so a simple retrieval function is enough to solve the problem (see below). On the other hand, if the level is a non-terminal one, all solutions that can be built in an underlying level have to be taken into consideration. In this latter case, after obtaining both sets of solutions we have to merge them.

Sign retrieval. In the paradigmatic phase of recognition, from a starting exploration state, within a particular level, lexicalized candidate signs contained in the level lexicon are matched against the remaining acoustic observations to obtain both a score and a new, more advanced exploration state. To enforce a real advance from one exploration state to the other, no empty significants are allowed in any sign.

The scores are then weighted by the lexicalization probabilities as kept in the lexicon, and the hypothesis containing the score, the new state and the lexicalized unit returned.

Level Delving and Phase Shifting. Although some applications may expect lexicalized signs in elocutions with a fairly good probability, most of the solutions to the recognition problem will have to be generated by a lower level by means of a sintagmatical generate-and-test procedure: given a sintagmatical starting exploration state and the sequence of observations, it will obtain a set of scored, syntactically built signs, and the sintagmatical exploration states they have reached.

To effectively use this delving scheme we need some stub or *phase-shifting* functions whose purpose is, first, to shift from the paradigmatic phase in the upper level to the sintagmatical phase in the lower level before calling the procedure, and second, to shift back the results from one phase to the other on return. Otherwise, the solutions from the lexicalized part and the constructed set would belong to different levels and phases and they could not be merged.

3.2. The Sintagmatic Phase: Level Delving

This phase takes its name from the fact that signs behave sintagmatically in the solution-building process within the level.

Purpose. This building procedure takes a level, a starting sintagmatical exploration state, an initial construction state and the acoustic observations still unaccounted for, and constructs a sequence of signs by advancing the exploration and construction states while consuming observations to the point where they could be phase-shifted into a unit in the upper level. Exploration states and construction states are so closely woven in the paradigmatic phase that we pair them under an *overall state*.

Terminating conditions. Given that we are in state *(s, x, c)*, in level *n*, with score s and exploration state *x*, where we have built so far a certain "portion" of a phrase *c* by accumulating signs onto it, if such exploration-construction state is a valid sign (after a validating function *g*), we must store and return it as a solution to the problem. We can decide as well whether if to stop the search process since a solution is reached or to go on exploring from this solution state: the latter is the solution adopted for top-n solutions mode.

Transition function. If we wanted to continue exploring, we would be interested in all possible signs that might follow the exploration-construction state. We would then try for each of the members of that set to advance the exploration and construction state one step forward. We can view this procedure as a non-deterministic, estochastic transition function in a search graph.

Recursion. We will call the set of possible signs following a particular exploration-construction state, its *follow-set*. These signs are in fact the solutions within this level to the problem of recognition given a particular exploration state (the present one), which is exactly an instance of the problem we are trying to solve, so we apply recursively the algorithm and obtain the solutions.

Once we have received such solutions we carry out the transitions from the present state to all the following states thus expanding this hypothesis story. Only the construction states need to be built because the exploration states have already been advanced in the recursive call.

The implementation. The search algorithm outlined above resembles closely the stack decoding algorithm when we allow for the use of built signs. We have a starting state (some exploration state with an empty construction state), a terminating condition (whether if the state designates a valid unit in the upper level), and an expansion or transition function dictating all possible transitions from a given state. We may use any of the schemes proposed by Paul ([3.], [4.]) to improve the original *A* algorithm, even at the cost of completion or admissibility.

Furthermore, if we are not attempting a time synchronous recognition and know the end of the elocution beforehand, we can use the minimum trained phonetic distance and the expected distance from each symbol in the grammar as non-related heuristics for a near-admissible, better-informed search.

3.3. Objection: Non-Recursive Jumps

Some linguistic processes seem to be out of the main recursive descent in the level hierarchy. For example prepositions almost always precede noun phrases, but the resulting preposition+noun phrase, almost always behaves like something in the word level (used as a word when building phrases). The same happens with relative pronouns and clauses, recategorized words like participles, etc. These phenomena challenge both a simple recursive descent technique, and the terminating proof of the algorithm.

While the termination is still maintained if we impose the criterion that each jump must advance the exploration state, non descending jumps request a more elaborate reorganization of the algorithm which implies *guidance*.

4. GUIDING THE PROCESS

In the state transition function, considering that only those construction states allowed by the grammar ought to be reached, many already built, costly solutions will be wastefully discarded. It would be better if the effort used in scoring and building went to units that could afterwards be kept in the hypothesis.

Targets. To avoid overgeneration and restrict paths in the search process we introduce the notion of *targets*. Targets are constraints on the solutions of the recognition function that can be stated before the process has started. In fact, the exploration of the input sequence of acoustic observations can be considered a "target" that has to be accounted for; however, our signs have a double domain, semantic targets also have to be taken into consideration, and, as no hypothesis is rejected on the grounds of acoustical matching, it falls to the semantic component to guide the search.

Target focusing. Too strict a constraint may completely rule out the desired solution, but too weak a constraint may not guide the search properly allowing overgeneration, so we must have a way to underspecify constraints and strengthen them along the recognition process. We call this *focusing* the target. To attain this, we adopt a linguistic framework that allows the *incremental* building of signs

and targets: traditional grammars are not suited for this purpose, but unification grammars are.

Path reduction. Targets have also two behaviours according to the phase they are used in the process: on one hand, paradigmatic targets constraint the number acceptable lexicalized solutions of the paradigmatic function (which accordingly reduces the number of paths explored in the sintagmatic one); on the other, targets in the sintagmatic phase guide the recognition process, first by predicting valid new targets and then by focusing the target already attained. Prediction is carried out just before recursively calling the paradigmatic function, to supply it with adequate targets for a new, lower level recognition: the grammar and the actual construction state dictate that not all signs are allowed in the follow-set for a particular overall state, but only those that will not fail to advance its present construction state.

Focusing. Focusing is carried out by the advance in the construction state brought about by accumulating the information supplied by the new signs in the semantics of the constructed sign.

Non-descending jumps. Finally, if targets have an indication of the level they must be solved in, they may be used to jump to other levels than the one directly underneath.

5. APPLICATION CHARACTERIZATION AND TRAINING

We have a DARPA-RM like application consisting of some 1000 sentences of which we use 600 for training and 400 for testing. The vocabulary is around 1k words.

The acoustical observations are phonetic strings obtained by a one-pass monophone, 3-state, discrete HMM decoder with 48 allophones, operating on 10 MEL frequency Cepstrum coefficients plus energy and their derivatives, using two codebooks with 256 centroids each. The cost matrix error model for the 48 allophones used was trained using a DTW match algorithm that dismissed all context influence.

The hardest part of the training is the lexical model training. The training function takes as its input a set tagged with the semantic categories of the upper level and each unit segmented into its constituents, and then proceeds through the following steps:

1. Design a new tag set that describes a lexical meaning classification tree for the units in this level and classify the constituents of the level corpus using this tree. For the time being, this is a computer assisted task.

2. Include in the level lexicon all lexicalized units and record their frequency. Index this lexicon with the classification tree so that both the set and the frequency interpretation of a meaning can easily be accessed.

3. Segment all non-lexicalized units into their constituent units belonging to the lower level (done semiautomatically) and recursively call this function to obtain the grammars that generate the segmentations in the lower level. Because these grammars all describe the derivation of a particular semantic tag in this level, each can also be indexed by its entry tag in the classification tree.

4. Build one grammar for each of the semantic tags used in the input (upper level) segmentation. A grammatical inference algorithm can be used for this task. Return the set of grammars obtained.

The inference algorithm used is ECGI [9.] which obtains estochastic finite automata; note however, that we have supplied our system with a recursive descent algorithm, so the power of the overall grammar may reach that of CFG.

6. REFERENCES

1. Bahl, L.R., Jelinek, F., and Mercer, R.L., "A Maximum Likelihood Approach to Continuous Speech Recognition", IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-5, March 1983.

2. Nilsson, N.J., *Principles of Artificial Intelligence*, Morgan Kauffman, San Mateo, Ca. EEUU, 1980.

3. Paul, D.B. "An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Estochastic Model", Procs. ICASSP-92, Vol. I, pp. 25-28, San Francisco, 1992.

4. Paul, D.B., "Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder", Procs. ICASSP-91, pp. 693-696, Toronto, 1991.

5. Paul, D.B., "A CSR-NL Interface Architecture", Procs. of the ICSLP-92, Vol. 2: pp. 1423-1425, Canada, 1992.

6. Cruse, D.A., *Lexical Semantics*, Cambridge University Press, UK, 1986

7. Miller, G. et al., "Five Papers on WordNet", official documentation of the WordNet Distribution.

8. Bunge, M., "La Metafísica, Epistemología y Metodología de los Niveles", in *Hierarchical Structures, Proceedings of the Symposium held Nov. 1968 at Douglas Advanced Research Laboratories*, American Elsevier Publishing Co., 1969

9. Rulot-Segovia, H.M., *ECGI: Un algoritmo de Inferencia Gramatical mediante Corrección de Errores*, Ph. D. Thesis, Universidad de Valencia, Spain, 1992.

7. ACKNOWLEDGEMENTS

This research was partially funded by a research grant "Formación de Profesorado Universitario" from the Spanish Ministry of Education and Science.

We would like to thank J.Colás-Pasamontes for supplying us with the acoustical model and front-end to generate the phonetic strings and J.Macías-Guarasa for his help with the DP techniques.