A DEMI-SYLLABLE BASED CONTINUOUS SPEECH RECOGNITION SYSTEM WITH HMMS AND SYNTAX-CONTROLLED WORD SEARCH

Walter Weigel

Lehrstuhl für Datenverarbeitung
Technical University of Munich

ABSTRACT - A system for recognizing continuous speech in a speaker-dependent mode is described, where demisyllables serve as basic processing units. The acoustic-phonetic decoding uses an explicit segmentation based on a pattern-matching technique and vowel-context-independent HMMs. The sentence recognition uses simplified word-HMMs and a Viterbi-algorithm. For the syntax-control a bottom-up and a top-down strategy are compared, achieving sentence recognition rate of up to 74%.

1. INTRODUCTION

The aim of this contribution is to give a survey of our speaker-dependent continuous speech recognition system with emphasis on the sentence and word recognition. The system is designed in order to use some knowledge about speech and it's structure. The recognition of fluently uttered speech is especially faced with two problems, namely the phenomenon of coarticulation and the word chaining. The latter refers to the large search space which is built by the combination of all words, whereby the number of words, the words themselves and moreover their boundaries are unknown. The term 'coarticulation' summarizes effects which are due to the principle of "economy of articulation". This means that in continuous speech the articulatory gestures of the intended phonemes are actually not reached or at least overlap in time. Therefore the acoustic representation of phonemes is highly dependent on their context, which should be borne in mind in recognition.

The presented system tries to exploit knowledge about speech in different ways:
-Firstly larger processing units than phonemes are used, namely demisyllables. They contain the coarticulatory effects between the phonemes of which they consist. Moreover demisyllables implicitly include phonological restrictions because not every possible sequence of phonemes builds a valid demisyllable.
- The syllabic structure of speech is regarded by localizing explicitly the syllabic nuclei. This furnishes a time-grid on which the acoustic-phonetic classification is based as well as the word and sentence recognition.
- The acoustic-phonetic decoding uses Hidden Markov Models which are trained in context (different vowel contexts) to get some independence of coarticulation effects caused by vowels.
- The word lexicon consists of so called word-models (pronunciation models) which cope especially with one coarticulation effect, namely vowel-elisions.
- Grammatical knowledge is exploited in order to reduce the combinatorial number of word sequences during the word and sentence recognition.

2. SYSTEM-STRUCTURE

The structure of the system is indicated by Fig.1. The first step is a prepocessing which is performed by a filter-bank. It results in 22 channels which model the human loudness sensation (Zwicker et al. 1979). The output is sampled every 10ms and digitized by a A/D-Converter with a resolution of 16 bit. Then follows the acoustic-phonetic decoding which itself consists of two ensuing modules. The first performs the localization and classification of the vowels or diphthongs, i.e. the syllabic nuclei are explicitly determined so that the segments between two succeeding time points always contain a final demisyllable and an initial demisyllable. The second module classifies the consonant clusters of these demisyllables by means of stochastic modelling. This acoustic-phonetic decoding is described in more detail in section 3. The word and sentence recognition is based on this sequence of phonetic labels furnished by the acoustic-phonetic decoding. Two different approaches were investigated:

- A bottom-up strategy which uses the Earley algorithm (Earley 1970) and a Context Free Grammar to control a Viterbi-algorithm proceeding from this label lattice.
- A top-down strategy where a modified A*-algorithm (implemented in PROLOG) fulfills the search and starts by exploiting a Definite Clause Grammar.

A detailed presentation of these alternatives is the subject of section 4 while section 5 gives some recognition results of this system.

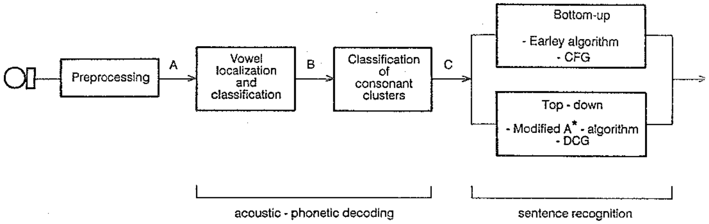
Preprocessing
A
Vowel localization and classification
B
Classification of consonant clusters
C
Bottom-up
- Earley algorithm
- CFG
Top - down
- Modified A* - algorithm
- DCG
acoustic - phonetic decoding
sentence recognition

Fig. 1 System structure

3. ACOUSTIC-PHONETIC DECODING

The first part of the acoustic-phonetic decoding localizes and classifies the vowels or diphthongs by using the short-time loudness spectra as input data (point A in Fig. 1). An example of these data is depicted by Fig. 2a. This is performed by computing a decision function where the syllabic nuclei result in local maxima as Fig. 2b shows. The dotted vertical lines indicate the centers of the vowels which build together with their labels the output of this module (point B in Fig. 1). The decision function itself is computed as a quotient of a special loudness contour (so called modified loudness because the high channels are substracted from the sum of the low channels) and a distance contour. The latter is the result of a pattern matching technique where reference vowel patterns are matched with the spectra ot the unknown sentence at each time point (10ms). The vowel or diphtong patterns were chosen by hand from fluent speech and are limited by a rectangular window to 60ms. If at each time point the minimum distance out of all matches is chosen and drawn over the time a distance contour is obtained. At those positions where vowels were uttered the contour shows obvious local minima because at least one of the reference patterns fits well. Using the quotient of both contours improves the quality and sharpness of the maxima. This method leads to an accuracy of up to 98,2%. Moreover the classification can be easily performed by choosing the label of the reference vowel which yielded the minimum distance at the indicated point. The recognition rate reached 88,5%.

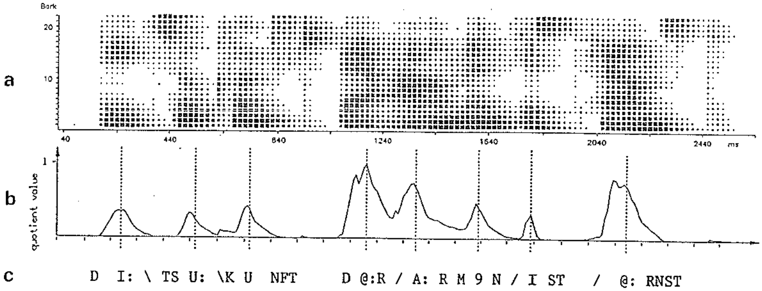

Fig. 2 Outputs of a) preprocessing, b) segmentation and c) classification

The time positions of the syllabic nuclei defines a time grid on which the second part of the acoustic-phonetic decoding bases. The intervall includes two demisyllables namely a final and an initial one. While their vowel parts have been already classified their consonant clusters are not known yet. Special care was taken about coarticulation caused by the vowel, which influences strongly the consonants . Hence each consonant cluster is represented by a discrete HMM which had been trained in different vowel contexts. This means in practice that during training each HMM has seen different demisyllables which consist always of the same consonant cluster but of different vowels. This method ensures a certain independence of the vowel context which can be demonstrated in experiments where the recognition rate is increased by about 10% in contrast to the use of consonant cluster models only (without training in context). In order to get an optimal classification of both consonant clusters included in the time intervall the localization of the syllable boundary is essential. Therefore an implicit segmentation was developed where all possible bounderies are tested during classification.

This module achieves recognition rates of up to 66,9% for consonant clusters where one has to bear in mind that each consonat cluster contains usually more than one phoneme (see Fig. 2c) so that the phoneme recognition is almost higher. This method and the investigations concerning the acoustic-phonetic module are described in detail in Weigel (1990).

4. WORD AND SENTENCE RECOGNITION

The word and sentence recognition uses for each lexicon word HMM's too (so called word models) which are drastically simplified (Ruske,Weigel 1986). The result of the acoustic-phonetic module consists according to point C in Fig. 1 of a sequence of labels in the fixed sequence: initial consonant cluster-vowel or diphtong-final consonant cluster. Therefore it is sufficient that the word models represent each lexicon word on this symbolic label. Such a word model is in fact a pronunciation graph, where especially vowel-elisions can be handeled by using skip-arcs over a whole syllable. The models are trained with an unsupervised method by applying the acoustic-phonetic module to sentences which include the words under consideration. In order to match the word models to the actual label lattice the wellknown Viterbi algorithm is used. As a measure of similarity between a symbol y of the word model and a symbol x of the sentence lattice we employ the a-posteriori probability p(y|x) which can be estimated by observing the confusions of the acoustic-phonetic stage. The question of interest is now how the single words can be recognized and how the huge search space spanned by all permutations of words with, additionally, different lengths and boundaries can be limited.

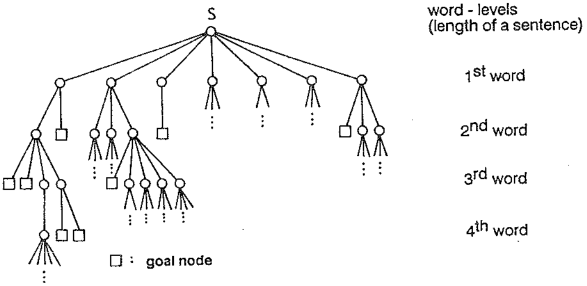

Fig. 3 Decision tree

Fig. 3 should give an idea of this decision tree where in general at each level (equal to the number of words) every lexicon word can follow if no constraints (e.g. syntactic or semantic ones) are applied. In order to to make use of syntactic knowledge two different strategies were investigated.

4.1 Bottom-up-strategy

The principle of the bottom-up-method is depicted in Fig. 4a. Starting with the result of the acoustic-phonetic decoding the symbolic lattice with the form initial consonant cluster-vowel or diphthong-final consonant cluster per syllable can be used to match the word models at each position. This can easily be fulfilled by the Viterbi algorithm implemented as a one-stage DP for connected word recognition. Here backtracking ensures that the optimal word sequence with respect to the used measure (in this case the a-posteriori probabilities) is obtained. Indeed, any syntactic restriction on the search space does not yet occur. As one solution the concatenation of the words during the Viterbi algorithm can be influenced by limiting the number of possible words during each word transition according to the syntax. Therefore it is necessary to know due to the processing direction of the Viterbi algorithm the allowed predecessor words of a given lexicon word. To get this information a Context Free Grammar (CFG) was used which describes by 40 production rules a subset of German affirmative sentences without relative clauses. The word lexicon size amounts to 132 words. An example of such a rule is

SATZ -> PP TSFV

which means that a sentence can consist of a prepositional phrase and a constituent with finite verb. An Earley algorithm exploiting these productions generates for a given word it's allowed successors according to the grammar. Since predecessor instead of successor words are necessary, a simple trick is used: By reversing the rules the so called "reversed grammar" yields the predecessors when the Earley algorithm is applied. This method corresponds to a word-pair grammar and has the advantage that it may be processed once and all predecessors can be stored before the real recognition task runs. As a disadvantage it does not guarantee recognition of completely correct sentences in the sense of the used syntax because the word transitions are local and the history of the sentence (i.e. the former words) is not considered. Thus the method is called "local Earley algorithm".

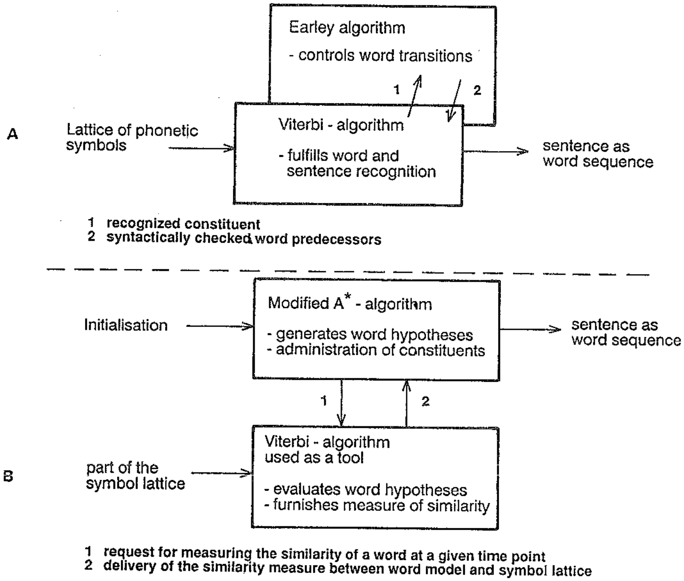
A
Earley algorithm
- controls word transitions
1
2
Lattice of phonetic symbols
Viterbi - algorithm
- fulfills word and sentence recognition
sentence as word sequence
1 recognized constituent
2 syntactically checked word predecessors
B
Initialisation
Modified A* - algorithm
- generates word hypotheses
- administration of constituents
sentence as word sequence
1
2
part of the symbol lattice
Viterbi - algorithm used as a tool
- evaluates word hypotheses
- furnishes measure of similarity
1 request for measuring the similarity of a word at a given time point
2 delivery of the similarity measure between word model and symbol lattice

Fig. 4 Syntax-control strategies

To avoid this a "global" version was developed where the predecessor words are determined indirectly. Here, at each time instant when a word transition can occur all lexical words which end at that point (with a final consonant cluster) are backtracked. Hence the history of each potential partial sentence is known and the Earley algorithm computes for this partial sentence the next words according to the grammar. These word successors are stored for each partial sentence. Looking for the allowed predecessors of a given word means now to search this given word in the stored successors. If it is found, this word can complete the partial sentence to which the store belongs. Now the best of all possible partial sentences can be chosen and the word transition is performed. This global Earley procedure is more complicated but increases the exploitation of the syntax because it considers the whole word sequence and thus insures absolutely correct sentences. Therefore the sentence recognition is significantly improved as will be shown in section 5.

4.2 Top-down strategy

As indicated by Fig. 4b an alternative solution to the sentence recognition problem was also investigated, namely a top-down strategy. There the recognition operates syntax-driven using a modified A*-algorithm. It performs the sentence recognition from left to right by generating word hypotheses, starting with all possible words at first position (level 1 in Fig. 3). These hypotheses are evaluated by the above mentioned Viterbi algorithm with the difference that it is only used as a tool. It computes the similarity measure for each hypothesis, matching it's word model to a given start point in the label lattice of the sentence to be recognized. Therefore it is not necessary to carry out any word transitions or even backtracking. The whole control of the word sequences is done by the modified A*-algorithm, which tries to find a solution to the decision tree. The usual A*-algorithm uses two cost-functions: One function g for the real costs of getting from the initial node to the current node. The second function h' is an estimation of the additional costs h of getting from the current node to a terminal node. In practice these costs follow directly from the applied similarity measure of the Viterbi algorithm. Hence the real costs g to a current node are as follows , where S denotes the number of syllables already processed from the start to this current node:

$$g = \sum_{i=1}^{S} \log p(y_i^I | x_i^I) + \log p(y_i^V | x_i^V) + \log p(y_i^F | x_i^F)$$

with x_i = classification result of syllable i, y_i = symbol of a word model
I = initial consonant cluster, V = vowel or diphthong, F = final consonant cluster

Similiarly the estimated costs h' to the terminal state can be computed because the acoustic-phonetic classification result of the not yet processed syllables is already known. Therefore instead of a symbol y of a word model the best case can be assumed where y is replaced by x itself. This assures that h' is always a lower bound and real costs h can only be equal to h' or will usually exceed h'. The formula is as follows, where N denotes the number of syllables of the sentence to be recognized.

$$h' = \sum_{i=S}^{N} \log p(x_i^I | x_i^I) + \log p(x_i^V | x_i^V) + \log p(x_i^F | x_i^F)$$

During investigations of a diploma thesis (Schiel, 1990) this usual A*-algorithm turned out to have a tendency of a breadth-first search until a first terminal node was found. The reason is that shorter paths are preferred to longer ones (which consist of more syllables) due to the addition of syllable costs. This can be avoided by dividing the costs g by the number of already processed syllables S:

$$g^* = g/S$$

Of course, this is no longer an optimal A*-algorithm. Hence another modification is necessary: After the first terminal node (i.e. one solution) is found another pass is carried out. The real costs g are again activated and used as a bound to check all other open nodes for lower real costs. So the optimality is retained although the whole procedure is drastically accelerated.

The grammar used is a Definite Clause Grammar which can be implemented relatively easily in PROLOG. It describes the same class of German sentences as the CFG. The number of rules accounts to 32 which is a lower number than for the CFG because the rules include attributes, e.g. for the coincidence of casus or numerus of the constituents. As an example may serve:

satz -> np(Num, Pers, Kas=nom), tsfv(Num, Pers)

This expression means that a sentence can consist of a nominal phrase where the case is nominative followed by a constituent with finite verb and that both have to coincide in number and person.

5 RESULTS

The system was tested with two different versions of 23 German sentences, which include the most important German vowels and consonant clusters. The versions differ in the date of the recording. For the training of the HMM's and the word models as well as for the vowel reference pattern generation 6 other versions of these sentences were used. The results are summarized in Tab. 1.

Method	Perplexity	Sentence-	Word-	Substi-	Omis-	Inser-	CPU-
		recognition		tutions	sions	tions	time
		[%]	[%]	[%]	[%]	[%]	[s]
Modified A*-algorithm	27	73,9	95,6	4,0	0	0	628
Earley alg. global	96	45,7	87,9	11,8	0,3	1,8	176
Earley alg. local	120	37,0	85,8	14,2	0	3,9	492
Viterbi alg. (no syntax)	132	34,8	86,1	13,9	0	3,9	12

Tab. 1: Recognition results

Regarding the sentence recognition it is obvious that the top-down method furnishes with about 74% correctly recognized sentences clearly the best result. However, the reason for this superior performance is not the method itself but the drastically lower perplexity of the Definite Clause Grammar. It's attributes reduce the search space much more than the Context Free Grammar can do. On the other hand it needs about a factor 4 of computing time compared with the global bottom-up method. To get an idea of the gain in performance the Viterbi algorithm as connected word DP was applied without syntax, where the recognition rate decreased to 35%. The expense of computing time which is necessary for the syntactic knowledge exploitation is shown by the comparison of the 12s CPU-time for the Viterbi algorithm alone and the 628s for the top-down method.

It is interesting, however, that the word recognition rate by the use of syntax did not increase as far as the sentence recognition rate. An analysis showed that sometimes relatively poor classification results, where false words fit well does guide the recognition with syntax in a false direction. On the other hand the Viterbi algorithm alone is absolutly free and yields the optimal solutions with respect to the classification result and the used similarity measure. If, however, one word was not recognized correctly the whole sentence can be judged as false which leads to low sentence recognition rate. Nevertheless it achieves about 86% word recognition rate.

ACKNOWLEDGEMENT

This work was supported by the Deutsche Forschungsgemeinschaft (DFG).

REFERENCES

EARLEY J.C. (1970) *An efficient context-free parsing algorithm*, Communications of the ACM, Vol. 13, No. 2, 94-102

RUSKE G., WEIGEL W. (1986) *Automatic recognition of spoken sentences using a demisyllable-based Dynamic Programming algorithm*, Intern. Congress on Acoustics, Toronto, A1-5

SCHIEL, F. (1989) *Syntaxgesteuerte Erkennung von gesprochenen Sätzen mit Methoden der Künstlichen Intelligenz*, Diplomarbeit, TU München, Lehrst. für Datenverarbeitung

WEIGEL, W. (1990) *Continuous speech recognition with vowel-context-independent Hidden-Markov-Models for demisyllables*, Intern. Conference on Spoken Language Processing, Kobe, 17.2 (in press)

ZWICKER E., TERHARDT E., PAULUS E. (1979) *Automatic speech recognition using psychoacoustic models*, Journ. Acoust. Soc. Am., Vol 65, 487-498.