Description of Training Procedure for AlfaNum Continuous Speech Recognition System

Niksha Jakovljevic, Darko Pekar

Abstract — **This paper presents training procedure for our continuous speech recognition system in Serbian, based on hidden Markov models. In this paper, we focus on solutions of the problem of insufficient training data, and improvements obtained by maximum likelihood train algorithm.**

Keywords — **Automatic speech recognition, hidden Markov models, maximum likelihood train algorithm and tying procedure.**

I. Introduction

THE automatic recognition system presented in this paper is speaker independent and proposed for telephone quality speech. It uses hidden Markov models (HMM) to represent phonemes. Phoneme characteristics depend on nearby phonemes, so usually modeling unit is context dependent phoneme, triphone. One of the main problems for this kind of system is to obtain sufficient training data. We propose solution based on phonetic similarity between phones, so called tying procedure.

Phoneme bounds are set by experts, so we have not had a need for complicated algorithms, which combat with the problems of roughly set phoneme bounds. Our early experiments with HTK tools proved this statement [1]. Our training procedure is based on K-means and Maximum Likelihood (ML) train algorithm.

The rest of the paper is organized as follows. In section II, a description of used corpus and features is presented. After that in section III, HMM modeling on phonetic level is described. One solution for obtaining sufficient observations per state is provided in section IV. General view on our training procedure is presented in section V. Experimental results for several variations in ML train procedure are provided in section VI, followed by conclusions in section VII.

This work was supported in part by the Ministry of Science and Environment Protection of the Republic of Serbia within the Project "Development of speech technologies in Serbian and their application in 'Telekom Srbija'" (TR-6144A).

Niksha M. Jakovljevic is with the Faculty of Engineering, University of Novi Sad, Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia & Montenegro; (phone: 381-21-4752999; fax: 381-21-450028; e-mail: jakovnik@uns.ns.ac.yu).

Darko J. Pekar is with the AlfaNum Ltd., Trg Dositeja Obradovica 6, 21000 Novi Sad, Serbia & Montenegro; (phone: 381-21-4750080; fax: 381-21-4750080; e-mail: darko.pekar@alfanum.co.yu).

II. Database and Features

The used corpus is a part of the SpeechDat database [2], containing only utterances spoken by male speakers. SpeechDat contains sentences, isolated words, digits, phrases, dates etc. spoken in Serbian. This corpora was recorded through telephone lines and sampled at 8 kHz with 8-bit A-law quantization. The training set contains 17752 utterances. The test set contains 620 utterances with 981 words. The training and test sets are disjunctive.

Speech signal is represented by cepstral coefficients, normalized energy, log energy and their first and second derivates. Features vector is divided into two streams. First stream contains 6 energy coefficients and second stream contains 36 spectral envelope coefficients.

III. Models

Basic modeling unit is a context dependent phoneme and/or subphoneme, called triphon. Besides standard phonemes with transcription in Serbian language, we use phoneme which corresponds to English phoneme /ə/, in our transcription 'Y'. Affricates and stops consist of two distinctive parts, closure (denoted by suffix 'o') and explosion (denoted by suffix 'e'), which are separately modeled. They are so called subphonemes. Set of Serbian vowels is extended to 10, making difference between stressed (denoted by suffix 's') and unstressed vowels. Silence and non-speech sounds, which are in the corpus, are modeled too. They are context independent.

Number of states per model is proportional to phoneme duration. All triphones, which represent the same phoneme in different context, have the same number of states. Number of mixtures per state depends on observations constellation in features space and is determined dynamically. During initial training maximum number of mixtures and minimum number of observations per mixture are specified.

Using triphones instead monophones leads to a very large set of models and relatively little training data for each triphone. All state distributions would be robustly estimated, if each state obtained enough observations. That could be achieved in two ways, first to extend training corpus, and second, to use additional observations of acoustically similar states during estimation of states with insufficient observations. Our choice is second solution. It is cheaper, but generates some suboptimal models (they are not so robust as in the case of having enough training data).

IV. Tying procedure

Tying procedure is realized on state level. How the states will be tied depends on which context is more important (left or right). For the first state and for all the states which are closer to the first than to the last state, more important context is left context, and for the last, central (if it exists) and all the states which are closer to the last then to the first state, more important context is right. More important context means that it stays permanent during searching for similar triphone. For example, take triphone S-A+M (phoneme A with left context S and right M) and suppose that it has not enough instances for training and phoneme A has 3 states. Its first state can be tied with only first state of phoneme A with left context S, and its central and last state with central and last state of phoneme A with right context M respectively. This principle is based on the fact that previous (left) phoneme have more influence on the leading states of the model, and successive (right) phoneme on ending states.

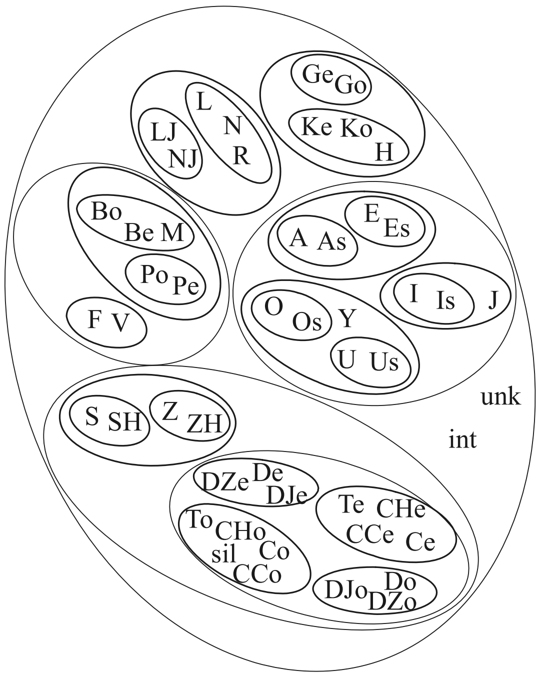

Fig. 1. Levels of similarity. Each set represent one similarity level.

Definition of phoneme similarity is based on our linguistic knowledge about their place and manner of articulation. Fig. 1 illustrates similarity levels. Each set represents one similarity level. If set includes several sets or sets containing several subsets, level of similarity is weaker. Level of similarity means that error, which is made during tying procedure, is smaller if observations of triphones, created using phonemes on lower level, are used.

Fig. 2 illustrates main idea of tying procedure. States of triphones with enough instances are estimated only on their observations, but their observations could be used during estimation of states with insufficient instances. For one triphone state, procedure starts on the lowest level and ends when the state gets sufficient instances for robust estimation or when all contexts are considered. It is important to note that obligatory context never changes even if after considering all possible phonemes as unnecessary context are not sufficient instances for training. Act of tying means creating information, which instances from the corpus will be used during estimation procedure for that state.

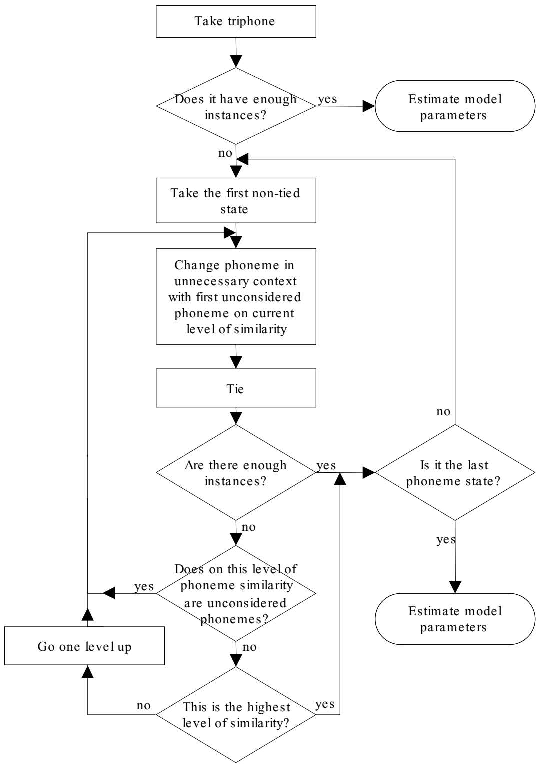
Take triphone
Does it have enough instances?
yes
Estimate model parameters
no
Take the first non-tied state
Change phoneme in unnecessary context with first unconsidered phoneme on current level of similarity
Tie
no
Are there enough instances?
yes
Is it the last phoneme state?
no
yes
Does on this level of phoneme similarity are unconsidered phonemes?
yes
Estimate model parameters
Go one level up
no
no
This is the highest level of similarity?
yes

Fig. 2. Flowchart of tying procedure

V. Training procedure

Training starts with tying procedure. After obtaining sufficient number of observations per state, K-means algorithm is used for initial estimation of model parameters. Input parameters for K-means algorithm are observations, minimum number of observations per mixture, and maximum number of mixtures per phoneme state. K-means algorithm finds optimal number of mixtures for each triphone state, by gradually increasing number of mixtures until average metric rise or maximum number of states is reached. To obtain robust models, we use ML train after K-means algorithm. ML train is last stage in our training procedure.

A. *ML train algorithm*

ML train algorithm expects good initial models and a sufficient data per state for robust estimation. Each state is described by weighted sum of Gaussian distributions:

$$b_j(o) = \sum_{m=1}^{M} c_{jm} N(o, \mu_{jm}, \sigma_{jm})$$

where: c_{jm} is weight of mixture m of state j, $N(o, \mu_{jm}, \sigma_{jm})$ Gaussian distribution with mean value μ_{jm} and variance σ_{jm}.

Every training sequence is segmented using a Viterbi alignment procedure. From that point, each state with associated observations is treated independently.

First, calculate probability that observation o_n belongs to mixture m of state j as:

$$p_{jmn} = \frac{c_{jm} N(o_n, \mu_{jm}, \sigma_{jm})}{\sum_{m=1}^{M} c_{jm} N(o_n, \mu_{jm}, \sigma_{jm})}$$

This probability is used for re-estimation of mixtures weights, means and variances as:

$$c_{jm} = \frac{\sum_{n=1}^{N} p_{jmn}}{\sum_{m=1}^{M}\sum_{n=1}^{N} p_{jmn}}$$

$$\mu_{jm} = \frac{\sum_{n=1}^{N} p_{jmn} o_n}{\sum_{n=1}^{N} p_{jmn}}$$

$$\sigma_{jm} = \frac{\sum_{n=1}^{N} p_{jmn} (o_n - \mu_{jm})(o_n - \mu_{jm})'}{\sum_{n=1}^{N} p_{jmn}}$$

Advantage of ML train algorithm over K-means algorithm, lies on soft distribution of observations per mixture in states. Each observation affects on all mixtures in the state, so distribution coverage of observations space is better. Detailed description of ML train algorithm can be found in [3].

VI. Experiments

We implemented several modifications of ML train algorithm. First modification is removing outliers. Outlier is observation, which does not belong to any mixture. Therefore, outlier has small value of probability to belong to any mixture in the state i.e. $c_{jm} N(o_n, \mu_{jm}, \sigma_{jm})$.

Fig. 4. presents WER of models, which are generated by ML train algorithm. WER of models, generated by K-means algorithm is 22.73%, is our referent point. Increasing of WER for small values of threshold comes from incorrect computation of models parameters. Outlier small logarithm probability for some mixtures stays unchanged (value is small but bigger then threshold) and became dominant in estimation, but for other mixtures it is eliminated. Presented results are obtained after only one ML train iteration.

Our first goal was to find value of probability threshold for outlier observations. Our algorithm first calculates logarithm of probability and than converts it into linear probability, threshold is set as logarithmic value. Minimal value for threshold is –250 because it is the smallest value for log probability. See fig. 3.

Second modification of ML train algorithm supposes that K-means algorithm finds optimal number of mixtures per state, so that number is fixed. To achieve that, mixtures weights, which are smaller than minimum weight, are set to minimum weight. Fig. 5 presents performances of this system for different values of outlier threshold.

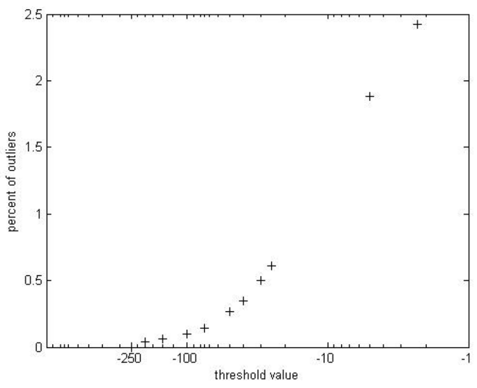

Fig. 3. Relationship between threshold value and percent of removed outliers

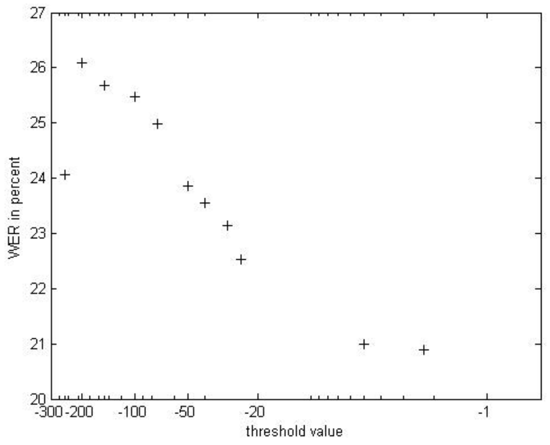

Fig. 4. Relationship between WER and threshold value after one step of ML train with removing outliers

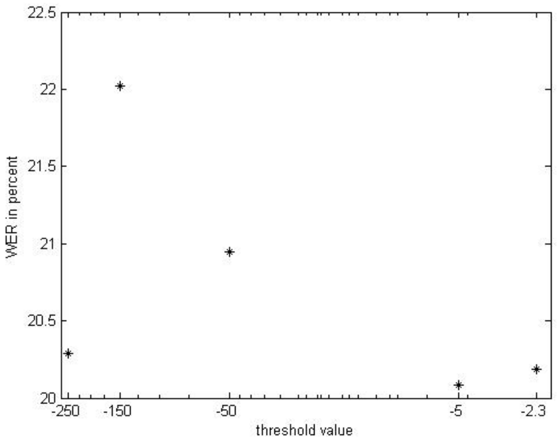

Fig. 5. Relationship between WER and threshold after one step ML train whit flooring small mixture weights and removing outliers.

We observed performance in smaller number of points then in the first variant, thus training procedure takes several hours and we already know trend between threshold and WER. This increasing of WER for threshold value –2.3 is the reason why we stop on that threshold value, although the percent of outliers in K-means is 3%. This raise of WER is indication that frames, which are not outliers, are expelled from estimation.

TABLE I : RESULTS FOR ML TRAIN WITHOUT REMOVING OUTLIERS

Iteration	WER [%]	number of false recognitions	number of insertions	number of deletions
0	22.73	129	53	41
1	24.06	135	43	58
2	24.57	135	32	74
3	24.97	140	36	69
4	24.36	139	38	62
5	24.36	139	37	63

TABLE II : RESULTS FOR ML TRAIN WITH OUTLIER THRESHOLD SET ON -5

Iteration	WER [%]	number of false recognitions	number of insertions	number of deletions
0	22.73	129	53	41
1	21.00	132	58	16
2	21.81	137	68	9
3	22.53	140	73	8
4	22.12	134	73	10
5	21.30	128	71	10

TABLE III : RESULTS FOR ML TRAIN WITHOUT REMOVING OUTLIERS BUT FLOORING SMALL MIXTURES

Iteration	WER [%]	number of false recognitions	number of insertions	number of deletions
0	22.73	129	53	41
1	20.28	117	36	46
2	19.16	102	29	57
3	21.92	91	26	98
4	19.88	89	26	80
5	18.55	95	27	60

TABLE IV : RESULTS FOR ML TRAIN WITH OUTLIER THRESHOLD SET ON -5 AND FLOORING SMALL MIXTURES

Iteration	WER [%]	number of false recognitions	number of insertions	number of deletions
0	22.73	129	53	41
1	20.08	126	60	11
2	20.39	122	71	7
3	19.37	113	70	7
4	18.04	102	69	6
5	18.14	101	72	5

After we had found optimal outlier threshold, we started iterative ML train procedure in several variants:

- ML train without removing outliers and its performances are tabulated in Table I.
- ML train with outlier threshold set on -5 and its performances are tabulated in Table II.
- ML train without removing outliers but flooring small mixtures. Its performances are tabulated in Table III.
- ML train with outlier threshold set on -5 and flooring small mixtures. Its performances are shown in Table IV.

The reason why we have continued to experiment on first variant, although it obtains degradation of performances (WER is 24.06%), is that this variant is based on unmodified ML train algorithm formulas.

First row (after zero iterations) represents performances of models, generated by K-means algorithm.

Including outliers in model re-estimation seriously degrades ML train procedure. Models generated by pure ML train algorithm have worse performances than initial ones (see Table I). Comparing results in tables II and III leads us to conclusion that outliers seriously degrade mixture weights. At this moment, we cannot explain the reason.

Even if increasing in average metric after each iteration step is achieved, WER has not constant decreasing. This is not a case for fourth variant of ML train.

When we made an error, we obtained interesting result. If during calculation of state probability in Viterbi algorithm, instead

$$b_j(o) = \sum_{m=1}^{M} \frac{c_{jm}}{\left(2\pi\sigma^2{}_{jm}\right)^{N/2}} \exp\{-0.5\frac{(o-\mu_{jm})(o-\mu_{jm})'}{\sigma^2{}_{jm}}\}$$

used

$$b_j(o) = \sum_{m=1}^{M} \frac{c_{jm}}{\left(2\pi\sigma^2{}_{jm}\right)^{N/2}} \exp\{-\frac{(o-\mu_{jm})(o-\mu_{jm})'}{\sigma^2{}_{jm}}\}$$

achieved performances were better. For models generated by K-means algorithm WER is 14.57%, but for models obtained after ML train iterations, the lowest WER is 15.49%.

VII. CONCLUSION

In this paper, we present our results with ML train algorithm. We found out that outliers are a big problem and further examination will be concentrated on recognizing outliers in ML train. We will also try to find the reason why the incorrect evaluation of state probability leads to better performances.

REFERENCES

[1] N. Jakovljevic, D. Pekar, “Performances comparison between ASR system made by HTK tools and AlfaNum CASR system ,”in *Proc.* ETRAN, H. Novi, 2003, pp 399-402

[2] N. Djuric, D. Pekar, Lj. Jovanov, “Structure of SpeechDat(E) database for Serbian, recorded over public telephone network,” in *Proc. DOGS 2002,*Becej, 2002, pp 57-60.

[3] L. Rabiner, B.H. Juang, Fundamentals of Speech Recognition. Englewood Cliffs, New Jersey: PTR Prentice Hall, 1993, pp. 350–352.