Effective Speaker Tracking Strategies for Multi-party Human-Computer Dialogue

Vladimir Popescu[1,2], Corneliu Burileanu[2], and Jean Caelen[1]

[1] Grenoble Institute of Technology, France
`{vladimir.popescu, jean.caelen}@imag.fr`
[2] "Politehnica" University of Bucharest, Romania
`cburileanu@messnet.pub.ro`

1 Introduction

Human-computer dialogue is already a rather mature research field [10] that already stemmed to several commercial applications, either service or task-oriented [11]. Nevertheless, several issues remain to be tackled, when unrestricted, spontaneous dialogue is concerned: barge-in (when users interrupt the system or interrupt each other) must be properly handled, hence Voice Activity Detection is a crucial point [13]. Moreover, when multi-party interactions are allowed (i.e., the machine engages simultaneously in dialogue with several users), supplementary robustness constraints occur: the speakers have to be properly tracked, so that each utterance is mapped to a certain speaker that had produced it. This is needed in order to perform a reliable analysis of input utterances [2].

Spoken human-computer dialogue systems can be seen as advanced applications of spoken language technology. A dialogue system represents a voiced and relatively natural interface between the user and a software application. Thus, spoken dialogue systems subsume most of the fields in spoken language technology, including speech recognition and synthesis, natural language processing, and dialogue management (planning).

A dialogue system involves the integration of several components, which generally provide the following functions [3]:

- **speech recognition**: conversion of an utterance (represented as a sequence of acoustic parameters), into a word sequence;
- **language understanding**: analysis of a word sequence in order to obtain a meaning representation for this sequence, in the dialogue context;
- **dialogue management**: system-human interaction control, as well as the coordination of the other components of the dialogue system;

- **task management**: interfacing of the dialogue management and language understanding modules, with the application domain for the tasks performed by the system;
- **answer generation**: computation of the sequence of words constituting the answer generated by the system, situating it in the discourse context represented by the dialogue history, and in the pragmatic context, represented by the relationship between user and machine, as well as by their social roles;
- **speech synthesis**: conversion of the text representing the system's answers, into an acoustic waveform.

Among these components, some (dialogue and task management, partially language understanding and answer generation) are language independent and (in part) dependent on the application domain, whereas others (speech recognition and synthesis) depend on the language, being (in principle) independent of the application domain. Thus, for the components in the first category, reuse in new languages is, to a great extent, possible, if the application domains are kept, whereas the components in the second category have to be developed for each new language, in a manner that is independent of the application.

The task of the speech recognition component in a spoken dialogue system consists in converting the utterance (in acoustic form) came from the user, into a sequence of discrete units, such as phonemes (sound units) or words. A major obstacle in accomplishing a reliable recognition resides in speech signal variability, which results from the following factors:

- *linguistic variability*: consists in the effects of several linguistic phenomena that the speech signal undergoes, such as phonetic co-articulation (i.e., the fact that the same phoneme can have different acoustic realizations in different contexts, determined by the phonemes neighboring the sound concerned);
- *speaker variability*: consists in the effects of inter- and intra-speaker acoustic differences; inter-speaker differences are determined by physical factors, such as the particular shape of the vocal tract, the age, sex or origin of the human subjects (the fact that a speaker may not be native in the language being used for communication); intra-speaker differences are determined by the fact that the same word can be uttered in several ways by the same speaker, according to her or his emotional or physical state, or to the pragmatic (and situational) context of the utterance - a word can be uttered more emphatically in order to stress a certain idea;
- *channel variability*: consists in the effects of the environmental noise (which can be either constant or transient) and of the transmission channel noise (e.g., microphones, telephone lines, or data channels - "Voice over IP").

The speech recognition component in a typical dialogue application has to take into account several additional issues:

- *speaker independence*: since the application is normally used by a wide variety of individuals, the recognition module may not be trained for one single speaker (or for a few speakers) supposed to use the system, as it is the case, for instance, in voice dictation applications. Thus, speech has to be collected from an acoustically representative set of speakers, and the system will use these data in order to recognize utterance came from (potential) users, whose voices were not used during training. This is why the performances of the speaker independent recognition process are generally poorer than for speaker dependent recognition.
- *size of the vocabulary*: the number of words that are "intelligible" to the the dialogue system depends on the application considered, as well as on the dialogue (management) complexity [10]. Thus, a strictly controlled and rather inflexible dialogue may constrain the user to a small vocabulary, limited to a few words expressing the options available in the system; yet, in more natural and flexible dialogues, the vocabulary accepted by the system can count for several thousands of words (for instance, the PVE - "Portail Vocal pour l'Entreprise" system, developed in France as a voice portal for enterprises, uses an about 6000 words recognition module [3]).
- *continuous speech*: the users are expected to be able to establish a conversation with the spoken dialogue system, using unconstrained speech and not, for instance, commands uttered in isolation. The issue of establishing the limits of the words is extremely difficult for continuous speech, since in the acoustic signal there is no physical border between them. Hence, linguistic or semantic information can be used in order to separate the words in users' utterances.
- *spontaneous speech*: since users' utterances are normally spontaneous and non-planned, there are generally characterized by disfluencies, such as hesitations or interjections (e.g., "humm"), false starts, in which the speaker begins an utterance, stops in the middle and re-starts, or extralinguistic phenomena, such as cough. The speech recognition module must be able to extract, out of the speech signal, a word sequence allowing the semantic analyzer to deduce the meaning of the user's utterance.

Dialogues between a computer and only *one* human partner are studied in a rather mature research field [10] and several commercial applications or systems exist in this respect, the situations where the computer is supposed to get involved in a dialogue with *several* humans at the same time, are still too little studied in a systematic manner. Several possibilities exist, towards multi-party human-computer dialogue:

- **multi-session** human-computer dialogue, where the machine gets involved in parallel dialogues with several humans; these dialogues are independent in that the speakers do not interact with each other and do not have access to the dialogues between the machine and the other speakers. This type of interaction is particularly interesting for situations involving concurrent access to a limited set of resources (e.g. meeting room reservation in a company); therefore, in this case there are several *classical* dialogues, on which the computer should maintain a coherent representation. Even if there is not a real multi-party dialogue, there is rather little work worldwide in this respect. For instance, current state of the art is represented by the PVE ("Portail Vocal pour l'Entreprise") system [3]. In this system, multiple sessions are handled, at the dialogue control level, through a game theoretic approach, where machine contribution sequences are evaluated via gains that are dependent, at the same time, on the task context (amount of resources, speakers' roles, etc.) and on the speech acts performed by the speakers.)

- **multi-party** human-computer dialogue, where the machine gets involved in *simultaneous* dialogues with several speakers; as in multi-session dialogue, the machine has to keep a coherent view on the dialogues; yet, there is a major difference in regards to the latter situation: in multi-party interaction, the dialogues are simultaneous, all the speakers being at the same place and having access to *all* speakers' utterances. This is why modeling (and *formalizing*) this type of interaction is particularly difficult. However, since around 2000 there is more and more (substantial) research work in this respect, trying either to study the portability of models designed for traditional dialogues, to multi-party dialogue [5], or to analyze multi-party dialogue corpora in order to determine the differences between traditional and multi-party dialogues [16], or even to give a formal account of particular aspects of multi-party dialogue (such as dialogue control) and concerning only some issues (such as the *shared* context between interlocutors) [8].

In multi-party dialogue, several speakers interact with the system, that thus has to be able to assign each human speech turn to a speaker identifier (in other words, the system has to figure out not only what has been said, but also *by whom* that has been said). Therefore, the speech recognition component in the dialogue system has to *track* the speakers as they produce their turns, and this should happen as fast as possible, so that the total processing time, for a request from a user, is as reduced as possible.

Thus, in this chapter we propose an algorithm for tracking speakers in dialogue; the procedure consists in an "on the fly" unsupervised MLLR (Maximum Likelihood Linear Regression) adaptation of acoustic models to speakers, where we derive a decision tree based on speech recognition scores at utterance level. This decision tree is then used to cluster utterances into speaker identities. An important point to emphasize is that the clustering process is performed *on-line*, for each new user utterance, but relying on previous utterances as well. The novelty of the method proposed in this chapter resides in that we use only "classical" unsupervised MLLR adaptation, but through a careful handling of confidence scores and decision-making based on these. Moreover, the computational simplicity (essentially assuming a top-down left-right traversal of a decision tree) is suited for dialogue applications where real-time operation is an important constraint. Concerning this latter aspect, several strategies for reducing speaker tracking time are studied: from a rather "naive" parallelization of recognition processes (for a speaker-independent system and several speaker-adapted systems), we further optimize the MLLR adaptation process *per se*, by combining the usage of phoneme-level regression classes [9] with a parallel running of the adaptation: MLLR is performed in parallel for the regression classes.

After a brief overview, in §2.1, of related research, concerning mostly *off-line* indexation of audio recordings of multi-party meetings, where real-time constraints thus do not apply, we review, in §2.2, the fundamentals of MLLR adaptation; then, in §3.1 we present the baseline speaker tracking algorithm, along with motivational background from psycholinguistics and corpus study; in §3.2 we propose several strategies for improving the runtime performance of the algorithm, via parallelization at several levels. Furthermore, in §4 we discuss several experiments performed with different versions of the speaker tracking algorithm, in the context of a book reservation multi-party dialogue application, in French and Romanian languages. The last section concludes the paper and proposes further enhancements.

2 Background

2.1 Related Work

As for the current state of the art regarding speaker tracking, most of the work is related to speaker segmentation and/or indexing of offline multimedia content (or recorded meetings); in that case, the task is eased by several points: meetings usually take place indoor, speakers have rather fixed positions, their number is rather constant throughout the meeting [18]. Thus, one of the few previous works on multi-party dialogue segmentation assigns turn taking likelihoods to a language model that reflects the nature of the conversations [13]; two algorithms run in parallel for speaker and speech content estimation on TV sports news. Hence, dialogue issues are not directly considered, since *enough* data is available offline and runtime constraints do not apply; moreover, in multi-party dialogues, where speaker changes occur in an unpredictable manner (it is only progressively that speech turns become available), hence statistically modelling speaker changes is much more complicated, if not impossible.

Another strand of research stems at performing both environment (i.e., noise features) adaptation and speaker adaptation and tracking, in pre-recorded meetings as well [18], [20], [21]. For example, in [20] and [21], an unsupervised speaker adaptation in noisy environments is performed, in order to segment recorded meetings, where usually several microphones (microphone arrays) exist and the relative positions of speakers with respect to the microphones can be exploited [12]. Usually, the approaches adopted in this context start from GMM (Gaussian Mixture Models)-based speaker identification systems, that are coupled with HMM (Hidden Markov Models)-based speech recognition systems [18], [20]. Concerning the microphone arrays approach, it usually relies on cross-correlations computed on signals coming from pairs of acoustic sensors [12]. However, none of these procedures apply to service-oriented dialogue applications, since they usually involve outdoor processing, where non-relevant speech signals exist as well and the geometry of the users positions with respect to the acoustic environment is not too much controllable [2], [11]. Moreover, there is another research strand that relies on multimodal input for speaker tracking, e.g. combining acoustics with vision [7].

However, the research closest to ours was pursued by Furui and colleagues [23], where one proposes an unsupervised, on line and incremental speaker adaptation methods that improves the performance of speech recognizers when there are *frequent* changes in speaker identities and each speaker produces a series of several utterances. Basically, the authors propose two speaker tracking methods, that are then applied to broadcast news transcription; first, an HMM based scheme is proposed, where the likelihood given by a speaker independent decoder is compared to the scores given by speaker adapted HMMs. The rationale behind this approach is that, for succeeding utterances from the same speaker, the speaker adapted decoder is expected to give a larger likelihood than the speaker independent HMM set; on the other hand, if the acoustic features differ from those of a previous (identified) speaker, then the speaker independent decoder is expected to yield a larger likelihood than the speaker adapted ones (unless the voice of the new speaker is very similar to the previous speaker's).

The adaptation is achieved using the MLLR method (see §2.2), but the new (adapted) mean vectors of the Gaussian mixture components in the states of the HMMs are updated so that overtraining (in the case of sparse adaptation data) is avoided, by linearly interpolating the adapted mean vector with the original (unadapted) mean. Moreover, the algorithm also tackles the situation where phonemes remain unadapted, because they are not acoustically realized in the adaptation data stream; this latter point is handled using vector field smoothing, whereby unadapted mean vectors are transferred to a new vector space (corresponding to the adapted HMM states) by using an interpolated transfer vector [23]. Then, speaker tracking supposes simply recognizing an utterance with a speaker independent system and with a set of speaker adapted decoders and comparing the likelihoods yielded by these processes. The second algorithm relies on GMMs for discerning the speakers, since computation is thus reduced; nevertheless, the initial speaker independent HMMs are adapted to speakers' voices as well, in order to improve utterance recognition performance.

Since in the goal of Furui and colleagues' research [23] was to improve the segmentation of multi-party (e.g., broadcast news) conversations, the correct speaker tracking rate was not a relevant measure, hence results in this respect are not reported; only word error rate improvements using speaker tracking are shown.

2.2 Fundamentals of MLLR Speaker Adaptation

In a generic service-oriented spoken dialogue system the speech recognition component is usually instantiated as a medium or high vocabulary speaker *independent* HMM based Viterbi acoustic decoder, followed an n-gram based linguistic model [6]. This approach is legitimate for classical dual-party dialogues, where the machine has only one (human) interlocutor. However, in multi-party dialogue, as the system has to figure out also *who* has uttered a certain speech turn, besides *what* has been said in that turn, speaker tracking has to be performed. An approach to this resides in *adapting* a speaker independent HMM decoder to the voices of the particular speakers taking part in the dialogue and expecting higher recognition (probability) scores for the speaker adapted systems, with respect to the speaker independent decoder. This is motivated by studies that have shown that a speaker adapted system usually has a word error rate around two times lower than the word error rate obtained with the speaker independent system [1], [4], [6]. Obviously, these higher recognition scores and lower error rates are achieved for speech signals produced by the speaker to whom the system was adapted; hence, for each new dialogue, new adapted systems have to be built, using a rather low amount of data (the length of a few utterances) and proceeding in an incremental manner (adaptation is further pursued as the speaker produces more utterances in dialogue).

As any data-driven HMM parameter estimation method (i.e., learning method), HMM adaptation can be *supervised*, when adaptation data are already labeled with textual information, or *unsupervised*, when adaptation data are not labeled. Moreover, when all adaptation data are available at once, *static* adaptation is performed; otherwise, if data become available along a time span, *incremental* adaptation is performed.

If static adaptation is performed in supervised manner, it can be done using the MLLR method, or the MAP (Maximum A Posteriori Probability) method. On the other hand, if incremental adaptation is performed, in supervised or unsupervised manner, it can be performed via the MLLR method [6]. Hence, we can follow two different approaches when adapting an HMM set to a speaker's voice:

- via MLLR method - for static or incremental adaptation, in supervised or unsupervised manner;
- using MAP criterion - only for static supervised adaptation.

Obviously, in multi-party dialogue applications where adaptation data becomes progressively available, incremental adaptation is the most appropriate choice, hence only the MLLR method can be used. In this case, the adaptation process essentially consists in computing a set of transforms which, applied to the HMMs due to be adapted, will reduce the mismatch between the HMMs and speaker dependent adaptation data. More specifically, MLLR is a model adaptation technique that estimates a set of linear transforms for the means and variances of the components in the Gaussian mixtures that model emissions at each transition between the states in the HMMs. The effect of these transforms stems at modifying the means and variances of these Gaussians, so that each state in the HMMs generates with maximum probability the adaptation data. However, it has been observed that, in practice, the most important performance improvements are obtained if only mean vectors are estimated, leaving the covariance matrices unchanged; modifying the latter parameters does not bring substantial improvements in recognition scores or error rates [9].

Denoting by $\mu = (\mu_1, ..., \mu_n)$ the n-dimensional mean vector of a component in a Gaussian mixture that models the output of one state in an HMM, and by $\overline{\mu}$, the re-estimated mean vector, using adaptation data stream s, the *transformation matrix* W is given by $\overline{\mu} = W \cdot \zeta$, where W is a $n \times (n+1)$ matrix, and ζ is the *extended mean* vector: $\zeta = (\omega, \mu_1, ..., \mu_n)^T$, where the upper index T denotes the transposition operation, and $\omega \in \{0; 1\}$ is an *offset* term; usually, the value $\omega = 1$ is preferred [22], in order to induce a non-negligible offset on the initial mean vectors.

The transformation matrix W can be computed in a manner that is akin to the linear regression method [22]. We denote by $s = (s_1, ..., s_T)$ a set of acoustic observations (an adaptation data stream), where each s_i, $i = 1, ..., T$ is a multidimensional vector (its dimension is given by the number of acoustic parameters used for characterizing a frame of signal - for instance, Mel cepstra); we denote by s_t the observation vector at moment t, by m, the index of a Gaussian component in a mixture, by μ_{m_j} the mean of the m_j-th component in the Gaussian mixture, by ζ_{m_j} the extended mean vector of μ_{m_j}, by Σ_{m_j}, the covariance matrix for the Gaussian of index m_j, and by $L_{m_j}(t)$ the *occupation* probability for the m_j-th component of a mixture, at time t (i.e., the probability that at time t mixture m_j models the output of the HMM, in the current state).

In order for the adaptation process to be robust enough with respect to data variability, it is sensible to compute distinct transformation matrices for different states in an HMM. However, computing a distinct transformation matrix for each state is often infeasible, due to data sparseness; this is why states can be *grouped*, so that they share a transformation matrix. One currently accepted criterion of state clustering (into *regression classes*) is the identity (or closeness) of the acoustic phenomena that these states account for [9], [6]: thus, the states that output, with maximum probability, the same phoneme type, are grouped in the same regression class, for which only one transformation matrix is built.

Thus considering that R Gaussian components of a mixture, forming a regression class denoted by the set of indexes $\{m_1, ..., m_R\}$, are adapted by computing the transformation matrix W_m, it has been shown [9] that the transformation matrix can be obtain from the equation: $\sum_{t=1}^{T} \sum_{r=1}^{R} L_{m_r}(t) \cdot \Sigma_{m_r}^{-1} \cdot s_t \cdot \zeta_{m_r}^T = \sum_{t=1}^{T} \sum_{r=1}^{R} L_{m_r}(t) \cdot \Sigma_{m_r}^{-1} \cdot W_m \cdot \zeta_{m_r} \cdot \zeta_{m_r}^T$. The occupation probabilities $L_{m_r}(t)$ can be expressed as $L_{m_r}(t) = P(\gamma_{m_r}(t)|s, HMM)$, where $\gamma_{m_r}(t)$ denotes the Gaussian of index m_r at time t, and HMM denotes the hidden Markov model currently adapted. This latter probability is usually computed using the *forward-backward* algorithm, from the adaptation data [6], [22]. Thus, the transformation matrix is computed in several steps: first, we denote the left-hand member of the equation above, by Z (since it does not depend on W_m); then, we define a new variable G_i with the elements $g_{jk}^{(i)} = \sum_{r=1}^{R} v_{ii}^{(r)} \cdot d_{jk}^{(r)}$, with $V^{(r)} = \sum_{t=1}^{T} L_{m_r}(t) \cdot \Sigma_{m_r}^{-1}$, and $D^{(r)} = \zeta_{m_r} \cdot \zeta_{m_r}^T$. Hence, the i-th row of W_m can be determined as $w_i^T = G_i^{-1} \cdot z_i^T$, where z_i is the i-th row of Z.

3 Speaker Tracking Algorithms

3.1 Baseline Adaptation Procedure

Outlook

The speaker tracking algorithm consists essentially in adapting a speaker-independent speech recognition system to each new utterance, then clustering these adapted systems into a more restrained set, denoting the speakers in multi-party conversation. The adaptation process is represented by an unsupervised MLLR adaptation of a set of speaker-independent HMMs, whereas the clustering is based on the top-down traversal of a decision tree involving the utterance-level log-likelihood scores obtained in speech recognition.

The inputs to the algorithm consist in:

- a set of speaker-independent trained HMMs (at a word, triphone or phoneme level); these are denoted by the system S_0;
- a set of acoustic features extracted from a test speech signal; such an utterance is denoted by ϵ_i, for the i-th user utterance in dialogue.

The output of the algorithm consists in an assignment of a speaker identifier to the input utterance. As for the intermediary information structures used, these consist in confidence scores obtained for each acoustic unit that occurs in an utterance; these scores are then averaged and the value obtained is denoted by σ_{0i}, for the i-th utterance and the system S_0. Another valuable set of intermediary data structures is represented by the MLLR transformation matrices [6], one matrix for each new adapted system. A system adapted to an utterance ϵ_i is obtained from S_0 via unsupervised MLLR using this utterance; hence, such a system consists in the original HMM set (S_0) together

Table 1. Characteristics of three vaudevilles

Play	"The Jackpot"	"The Railroads"	"The Martin Prize"
n°. of scenes	52	42	42
n°. of characters	17	18	8
n°. of main characters	6	6	4

with the transformation matrix for the MLLR adaptation to ϵ_i, and is denoted by S_{ai}.

Sociolinguistic evidence prove that spontaneous multi-party dialogues tend to involve at most 5-6 speakers [2]; for a greater number of speakers, we tend to have several independent dialogues, although the interlocutors might still share the same environment (table, desk, etc.). In order to test these evidence, we have considered a corpus of multi-party dialogues; the data consists in three vaudevilles written in the 19th century by Eugène Labiche were considered (in French language): "La Cagnotte" ("The Jackpot"), "Les chemins de fer" ("The Railroads"), and "Le prix Martin" ("The Martin Prize")[3].

Some relevant characteristics of these three plays are the number of scenes for these plays, their total number of characters, as well as the number of main characters in each play; we add that each scene has a number of speech turns varying from 2 to around 200, for a number of characters varying from 2 to 10. These characteristics are summarized in Table 1.

In order to provide a subtler characterization of this multi-party human-human dialogue corpus, we show in Table 2 the "raw" number of dialogues, with respect to the number of turns and dialogue partners. In Table 2 we can see that the distribution of the number of dialogues, with respect to the set of speakers and to their size is rather uneven: most of the dialogues have less than 50 speech turns, produced by less than 7 speakers. That is, our data presents an even distribution for dialogues of at most 50 turns, where at most 6 speakers participate. This situation is in accord with results from sociolinguistics, where, in social reunions, people tend to cluster in interacting groups of 4 to 6 individuals [19]. Moreover, dialogues tend not to be very long, namely, they usually contain less than 50 speech turns. However, there are a few longer dialogues, of around 80 speech turns, where 3 or 4 speakers are involved. These elements provide, in our opinion, valuable guidelines concerning the limits of multi-party dialogues, in terms of number of turns and participants: the machine should thus handle mostly conversations where at most 6 speakers are involved (including itself), whereby at most 50 speech turns are produced.

One last remark finds its place here, namely that, summing over the dialogues in Table 2, we see that a number of 133 is obtained; however, summing over the scenes in Table 1, a number of 136 is obtained. Nevertheless, we have

[3] The electronic versions of these plays were downloaded from http://fr.wikisource.org.

Table 2. Number of dialogues, according to their size and number of interlocutors

N°. of characters	2	3	4	5	6	7	8	9	10
N°. of lines									
2-10	**11**	**10**	**2**	**2**	0	0	0	0	0
11-20	**13**	**10**	**2**	**2**	**1**	0	0	0	0
21-30	**10**	**4**	**3**	**5**	**1**	0	0	0	0
31-40	**4**	**3**	**4**	**2**	**6**	**1**	0	**1**	0
41-50	**1**	**3**	**3**	**3**	**1**	**1**	0	0	0
51-60	**1**	**1**	0	0	**2**	**1**	0	**1**	0
61-70	0	0	**1**	0	**1**	**2**	0	0	0
71-80	0	**2**	**2**	0	0	0	**1**	0	0
81-90	0	0	0	**2**	0	0	**1**	0	0
91-100	0	0	0	0	0	0	0	0	0
101-110	0	**1**	0	0	**1**	0	0	0	**1**
111-120	0	0	0	0	0	0	0	0	0
121-130	0	0	0	0	**1**	0	0	0	0
131-140	0	0	0	0	0	0	0	0	0
141-150	0	0	0	0	0	0	0	0	0
151-160	0	0	0	0	0	**1**	0	0	0
161-170	0	0	0	0	0	0	0	0	0
171-180	0	0	0	0	0	0	0	0	0
181-190	0	0	0	0	0	0	0	0	0
191-200	0	0	0	0	0	**1**	0	0	0

previously stated that each scene is assimilated to one multi-party dialogue situation. The difference between the two counts stems from the fact that some scenes (namely, 3) are *monologues*, hence not considered in Table 2.

Hence, the speaker tracking algorithm adopts different strategies, regarding whether this number (denoted by $\overline{L}$) of speakers has been reached or not: while the number of speakers detected is inferior to $\overline{L}$, the procedure will create a new adapted speech recognition system for each input utterance. When the number of speakers reached $\overline{L}$, the algorithm will perform several supplementary tests before creating a new adapted speech recognition system. However, $\overline{L}$ is an input parameter for the algorithm; Wizard-of-Oz dialogue simulations [3] or corpora investigations (as shown above) can assess the scale of the dialogue (in terms of maximum number of speakers) and, by consequence, provide an empirical, application-dependent value for $\overline{L}$.

Decision-Making

As stated before, the speaker tracking algorithm uses the information structures described above by constructing a fixed decision tree and then traversing it accordingly. The tree is specified offline, whereas its traversal depends on the confidence score obtained in recognizing the input utterances; thus, the procedure goes as follows (numbers indicate successive steps, while letters mark alternative paths):

1. start with the speaker-independent speech recognition system S_0, a total number of utterances $N \leftarrow 0$ and of speakers $L \leftarrow 0$; specify a maximum number of speakers $\overline{L}$ and an *offset* Δ (the number consecutive of input utterances where no new speaker is detected);
2. for an input utterance ϵ_1:
   2.1 perform unsupervised MLLR of S_0 on ϵ_1, obtaining the adapted system, S_{a1};
   2.2 perform speech recognition of ϵ_1, with both S_0 and S_{a1}; two recognition scores σ_{01} and σ_{a11} result, respectively; from the definition of MLLR, we should have that $\sigma_{a11} > \sigma_{01}$; we mark that ϵ_1 has been produced by the speaker l_1;
   2.3 $N \leftarrow N + 1$, $L \leftarrow L + 1$;
3. for a new utterance ϵ_2:
   3.1 perform unsupervised MLLR of S_0 on ϵ_2, obtaining the adapted system S_{a2};
   3.2 perform speech recognition of ϵ_2, with the three systems S_0, S_{a1} and S_{a2}; three recognition scores are obtained, respectively: σ_{02}, σ_{a12} and σ_{a22}; we can have one of the following possibilities:
      (a) if $\sigma_{02} > \max(\sigma_{a12}, \sigma_{a22})$ then, from the definition of MLLR, we have an error;
      (b) else, if $\sigma_{a12} > \max(\sigma_{02}, \sigma_{a22})$, then we have an error as well;
      (c) else, if $\sigma_{a22} > \max(\sigma_{02}, \sigma_{a12})$, then ϵ_2 has been produced by a new speaker, l_2, so that $l_2 \neq l_1$; by consequence, $L \leftarrow L + 1$;
      (d) else, $\sigma_{a22} \approx \sigma_{a12} > \sigma_{02}$; then ϵ_2 has been produced by the same speaker as ϵ_1, that is, l_1; by consequence, L remains unchanged and S_{a2} is discarded;
   3.3 $N \leftarrow N + 1$;
4. repeat step 3 until $L = \overline{L}$ or L remains unchanged for a number of utterances equal to Δ;
5. if $L = \overline{L}$ or L unchanged for Δ consecutive utterances, for a new utterance ϵ_m, assuming that we have $1+W$ speech recognition systems, S_0, S_{a1}, ..., S_{aW} built as above, perform speech recognition on ϵ_m, with all the $1+W$ systems, obtaining the scores: σ_{0m}, σ_{a1m}, ..., σ_{aWm}; we can have one of the following possibilities:
   (a) if $\sigma_{0m} > \max_{i=1,\dots,W}(\sigma_{aim})$, then ϵ_m has been produced by a new speaker, l_{L+1}, different from the already detected L speakers; in that case, we perform unsupervised MLLR of S_0 on ϵ_m, obtaining a new system $S_{a(W+1)m}$; we perform $L \leftarrow L + 1$ and $N \leftarrow N + 1$ as well (actually, $m = N + 1$);
   (b) else, if there exists an $i \in \{1, ..., W\}$ so that $\sigma_{aim} > \max(\sigma_{0m}, \sigma_{a1m}, ..., \sigma_{aWm})$, then ϵ_m has been produced by the emitter of a preceding utterance ϵ_i, with $i < m$; in this case, L remains unchanged and N gets incremented by one, to obtain m;
   (c) else, if there exists a k in $\{1, ..., W\}$ such that $\sigma_{akm} \approx \sigma_{0m}$, then we have an error, from the definition of MLLR;

```
 (d) else, if there exist $j$ and $k$ in $\{1, ..., W\}$ so that $j \neq k$ and $\sigma_{ajm} \approx \sigma_{akm} > \max(\sigma_{0m}, \max_{t \neq j,k}(\sigma_{atm}))$; in this case:
     5.1 perform unsupervised MLLR of $S_{aj}$ and $S_{ak}$ on $\epsilon_m$, obtaining the systems $\tilde{S_{aj}}$ and $\tilde{S_{ak}}$, respectively;
     5.2 perform speech recognition with $\tilde{S}_{aj}$ and $\tilde{S}_{ak}$ on the utterance $\epsilon_m$; the scores $\tilde{\sigma}_{ajm}$ and, respectively, $\tilde{\sigma}_{akm}$ are obtained; in this point, two situations are possible:
         (a) if $\tilde{\sigma}_{ajm} \approx \tilde{\sigma}_{akm}$ and $\tilde{\sigma}_{ajm} \geq \sigma_{ajm}$ and $\tilde{\sigma}_{akm} \geq \sigma_{akm}$, then $S_{aj} \equiv S_{ak}$ and $\epsilon_m$ has been produced by the emitter of $\epsilon_j$ *and* $\epsilon_k$; in this case, discard $S_{ak}$ and $L \leftarrow L - 1$, $N \leftarrow N + 1$;
         (b) else, if $\tilde{\sigma}_{ajm} > \tilde{\sigma}_{akm}$ or $\tilde{\sigma}_{akm} > \tilde{\sigma}_{ajm}$, then denote by $j_0$ the index of the maximal score: $j_0 = \arg\max(\tilde{\sigma}_{ajm}, \tilde{\sigma}_{akm})$:
             (i) if $\tilde{\sigma}_{aj_0m} \geq \sigma_{aj_0m}$, then $\epsilon_m$ has been produced by the same speaker as $\epsilon_{j_0}$ (the utterance used to obtained the system $S_{aj_0}$); in this case, keep $L$ unchanged and $N \leftarrow N + 1$;
             (ii) else, $\epsilon_m$ has been produced by a new speaker, which is neither the producer of $\epsilon_j$, nor the producer of $\epsilon_k$; in this case, $L \leftarrow L + 1$ and perform an unsupervised MLLR of $S_0$ on $\epsilon_m$;
6. while there is an input utterance, go to step 5;
7. for $i$ from 1 to $N$, return the identifier of the speaker that produced utterance $\epsilon_i$.
```

In this algorithm, the decision tree is constituted by the "if" alternatives at steps 3.2, 5, 5.(d)5.2, and 5.2(b); the depths of the leaves are given by the nesting levels in the algorithm. The bottom-up traversal of the tree is inherently given by the nestings in the algorithm, whereas the left-right traversal is given by the order of the clauses: first, the loop in steps 3-4 is executed, then, the loop in steps 5-6. In Figure 1 this tree is represented, marking by "#" the first decision point, between the two speaker tracking strategies in steps 3-4, respectively 5-6; dotted arrows indicate the flow of the algorithm and the continuous lines mark alternative possibilities (the intersection of a set of such lines is a decision point). The rest of the symbols mimic those used in the specification of the algorithm; the tree should be read top-down, left-right.

As for the reliability of the algorithm, one objection might be that the variations in recognition scores can be induced by the variations in the content of the utterances used in adaptation or in recognition. However, this apparent problem is reduced by the fact that each comparison is performed between scores obtained on the *same* utterance, although the systems used for this might or might not have used the same utterance in training (or adaptation). If we had compared two scores obtained with two systems that had not used the same utterance set in training or adaptation, we could have had results "corrupted", i.e. the scores reflect rather the differences in utterances than the differences in speakers. The answer to this is that the scores are computed by averaging at the utterance level the scores obtained for each acoustic unit (e.g.

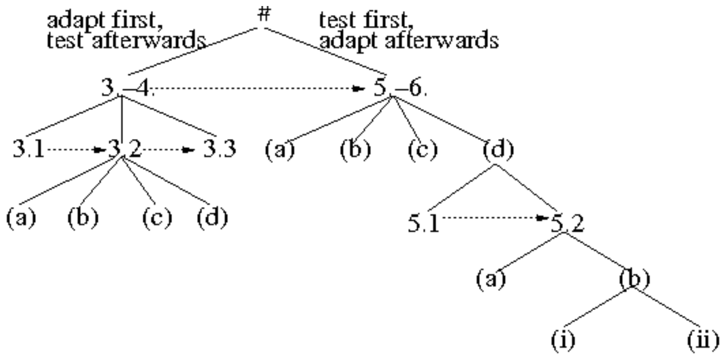

Fig. 1. Recognition score-based decision tree.

word, triphone, phoneme); hence, the scores that are compared might depend only on the particular distribution of the acoustic units within the utterance, being independent of the length of that utterance. However, even in this case, if the speaker-independent system is well trained, the scores for the individual acoustic exhibit low variances from one unit to another. Therefore, if this variance is inferior to the difference between scores obtained on utterances from different speakers, then the problem is alleviated. A discussion in this respect is provided in §4.2.

The parameter Δ (called “offset”) represents the maximum number of consecutive utterances where no new speaker is detected, before the algorithm adopts the “less expensive” strategy, starting from step 5. Its value can be empirically chosen, but a reasonable value is just the number L of detected speakers.

Concerning the complexity of the algorithm, expressed in terms of the number of speech recognition processes (performed as Viterbi decoding [6]), the number of MLLR adaptation processes, the number N of utterances in dialogue, the “expected” number of speakers ($\overline{L}$) and the offset Δ, an estimate is provided here. Thus, considering the worst-case scenario where all the branches in the tree are visited and the values of L and N are as high as possible (i.e., $L = \overline{L}$ from a certain input utterance on, and N limited - between steps 3 and 5, to $\overline{L} - 1 + \Delta$, and $\Delta = \overline{L}$), and denoting by τ_{MLLR} the average time needed for a MLLR adaptation process, by τ_{ASR}, the average time needed for a speech recognition process, and by τ_{CMP}, the time required for a comparison, we obtain that the execution time of the algorithm has an expression of the form (α, β, α' and β' indicate constant non-zero real numbers): $T = \tau_{MLLR} \times (\alpha\overline{L} + \beta N) + (\tau_{ASR} + \tau_{CMP}) \times (\alpha'\overline{L}^2 + \beta'\overline{L}N)$.

Therefore, the algorithm is quadratic in $\overline{L}$ and linear in N, for a specified $\overline{L}$ which is a constant for a running instance of the algorithm.

As for comparisons with previous work, we see that, according to Murani and colleagues [13], including speaker change information in the language model used in recognition improves recognition accuracy with around 2 % and speaker tracking performance with around 7 - 8 %; however, as already shown before, in multi-party dialogues such a model cannot be computed, because speakers' involvment in dialogues is not an already available information that could be used in training an n-gram. More interesting comparisons are possible with the work of Furui and colleagues: for instance, although in our baseline speaker tracking algorithm we do not use phoneme-level regression classes, performance is still acceptable, because in our procedure we have a supplementary step: when recognition scores obtained with two or more speaker adapted decoders are identical (up to a slight difference, less than 5 % of the average value of the scores), then these systems are *further* adapted to the current input utterance and the variations of the recognition likelihoods are studied (step 5.(d)5.1).

3.2 Performance Improvements

Algorithm Parallelization Strategies

Several strategies can be pursued for improving the runtime performance of the speaker tracking process. One such way resides in building a GMM for each speaker, and relying on these models in order to discern among interlocutors; this has been pursued by Furui and colleagues [23], but has the main disadvantage of reducing the accuracy of the speaker models, if two or more speakers in dialogue have very similar voices. This is why we have followed a different approach in improving the runtime performance: we studied possibilities of *parallelizing* the speaker tracking algorithm at several levels. This is motivated by the fact that nowadays multi-core computers or even clusters become readily available.

Thus, a first and very important parallelization step (undergone, among others, also by Furui and colleagues [23]) consists in simultaneously performing speech recognition of an input acoustic stream, with the speaker-independent and the speaker-adapt*ed* systems. Hence, we obtain a relatively steady runtime performance improvement in that computation time remains relatively constant with respect to the number of speakers (hence, to the number of speaker-adapted systems) in dialogue; this obviously becomes more important as the number of speakers increases. The runtime gain results in that, instead of adding the recognition times for the speaker-independent and speaker-adapted systems, we divide the sum of these times by the number of processors available (and if this number is approximately equal to the number of speakers - usually, around 4-5 speakers, as pointed out in §3.1, then the computation time practically does not increase when a new speaker gets involved in dialogue).

However, there is another point where performance can be improved, especially until the number of dialogue participants stabilizes, namely the actual MLLR adaptation process. Even if MLLR is not performed anymore, once the number of speakers in dialogue stabilizes (on average, to no more than 5 speakers), its runtime costs are important for the first speech turns in dialogue, where more and more speakers become involved. Actually, there are several dialogues where the number of speakers stabilizes towards the *end* of the conversation, thus MLLR could actually be performed at several points in dialogue, not only in its beginning. Moreover, a once involved speaker might leave the dialogue, but in some situations (e.g. when her/his voice was very similar to another, still active, speaker's voice), such an "on-leave" speaker has to be ruled out through further MLLR adaptation processes. This is why we have tried several MLLR parallelization strategies, finally adopting the most efficient one, as described in the next section.

Parallelizing the Adaptation Process

We have shown in previous studies [14], [15] that in parallelizing a data-driven HMM parameter estimation process, several strategies can be, in principle, adopted.

For example, we could try a *program-level* parallelization, where the actual sequential code that implements a parameter estimation procedure (e.g. Baum-Welch re-estimation, Viterbi alignment or MLLR estimation) is parallelized, either in an *algorithm-independent* way, via classical program optimization techniques such as *loop unrolling* or *multisampling*, or in an *algorithm-dependent* manner, by taking into account the inner working of the actual procedure (e.g. as in Ravishankhar's efficient Viterbi decoders in the Sphinx system [17]); in all these cases, all available data is used by each parallel program instance.

On the other hand, we can shift the focus from programs to data and thus follow a *data-level* approach, where the HMM parameter estimation procedure is parallelized by *distributing* the available data among several processors and then combining the results obtained; this can be realized either in an *algorithm-independent* way, where the details of the actual algorithm are not taken into account, but its (partial, since on partial data) results are combined or compared, as in [15], or in an *algorithm-dependent* manner, where the partial results are combined in a way that explicitly takes into account the inner workings of the algorithm, such as in HTK's parallel Baum-Welch HMM training [22]. In this strand, the algorithm-dependent (data-level) approaches have been shown to be the most efficient ones (as compared to the algorithm-independent data-level ones, that are, instead, more general).

Usually, in parallelizing such a procedure, we first investigate data-level approaches (first, algorithm-independent, then, algorithm-dependent), and only then we investigate program-level approaches (first, algorithm-dependent and, at last, algorithm-independent). This is motivated by the need to gradually modify a baseline, sequential procedure, that should be available as a starting point; this is of course necessary for proper bug tracking and development control.

Concerning MLLR parallelization, there is, to our knowledge, no reported research. This can be explained by two facts: first, MLLR is a rather new technique [9] and parallelizing it did not seem motivated by the (usually) *off-line* adaptation process; secondly, MLLR has been scarcely or not at all applied to actual multi-party dialogue systems, where real-time operation is of paramount importance. Hence, in trying to apply the strategies shown above, the most interesting ones are those that follow the data-level approach, since results are not affected by potential (hidden) parallelization bugs: the actual parameter estimation program remains *unchanged*, for example regarding a sequential baseline that has already been validated. We thus restricted our attention to data-level approaches to parallelizing MLLR: an algorithm-independent approach is not very useful, since, given that generally adaptation data is already rather scarce, the chances that saturation is achieved with less data (than that available) is very weak [9], which is not the case, for example, in Baum-Welch training, where indeed, data that do not further help the parameters converge can be thus detected and actually not used in training [15].

For the reasons above, we investigated the possibility of parallelizing MLLR in a data-level algorithm-*dependent* approach, where we cluster phoneme occurrences in a data stream, into regression classes, one class for each phoneme type that occurs in a (continuous) stream of adaptation data. Thus, a given stream s, of a certain speaker, contains the words $w_1^{(s)}, ..., w_N^{(s)}$, as recognized with the speaker-independent HMM system. Each word $w_i^{(s)}$ contains several phoneme occurrences (tokens), $p_1^{(is)}, ..., p_M^{(is)}$. Assuming that a word-level HMM modelling with a variable number of states per HMM (proportional to the number of phonemes in the word) is used, for each phoneme $p_j^{(is)}$ we have a number of states in the HMM of word $w_i^{(s)}$; otherwise, if phoneme-level HMM modelling is used, for each phoneme $p_j^{(is)}$ we have the model HMM_{ϕ_k} if $p_j^{(is)} \in c_{\phi_k}$, for phoneme type ϕ_k, $k = 1, ..., P$.

We compute a MLLR transformation matrix for each phonemic regression class; all phoneme tokens $p_j^{(is)}$ take values in the set $\{\phi_1, ..., \phi_P\}$ of phoneme (types) in a given (e.g. Romanian or French) language. The MLLR adaptation process computes a transformation matrix for each Gaussian (mixture) in each HMM state. We *cluster* the states that belong to the same phoneme in an adaptation data stream. For adaptation data stream s we have $p_j^{(is)}, j = 1, ..., |w_i^{(s)}|, i = 1, ..., |s|$, where $|w_i^{(s)}|$ denotes the number of phoneme occurrences in word $w_i^{(s)}$, and $|s|$, the number of word occurrences in stream s, and for each $p_j^{(is)}$ we have three states in a word-level HMM.

Now, denoting by $c_{\phi_k}^{(s)}$ the set of phoneme tokens in stream s that are equal to the phoneme type ϕ_k ($c_{\phi_k}^{(s)} = \{p_j^{(is)} : j \in \{1, ..., |w_i^{(s)}|\}, i \in \{1, ..., |s|\} | p_j^{(is)} \equiv \phi_k\}$), we can compute the time taken by several adaptation configurations:

(a) *Baseline sequential*

In this situation, no regression classes are used, all adaptation data are used to estimate only one transformation matrix; data are expressed as $s = s_1, ..., s_K$, where $(s_i)_{i=1,...,K}$ is an acoustic (observation) vector (for example, of MFCC cepstral coefficients). Thus, we use $\{s_1, ..., s_K\}$ to perform MLLR for all (HMM) states of all phoneme tokens $p_j^{(is)}$ in stream s. Denoting by τ the time needed for computing one transformation matrix, using one acoustic frame of adaptation data, in this process we have a time of $K \times \tau$ for computing the unique transformation matrix for all the states in the composite HMM of stream s.

(b) *Sequential, with regression classes*:

In this setting, a regression class is constructed for each phoneme type that occurs instantiated in stream s; in each such class we cluster all phoneme tokens in the words in s, that are identical, to a certain phoneme type. To make things clearer, we assume that each acoustic observation (vector) corresponds to the acoustic realization of one phoneme, and we denote by $s^{(k)}$ the set of acoustic frames that correspond to phoneme occurrences in $c_{\phi_k}^{(s)}$: $s^{(k)} = \{s_{i_1^{(k)}}, ..., s_{i_Q^{(k)}}\}$, with $\{i_1^{(k)}, ..., i_Q^{(k)}\} \subset \{1, ..., K\}$. Assuming that in stream s only $P_s' \leq P$ phoneme types are acoustically realized, we have that $\sum_{k=1}^{P_s'} |s^{(k)}| = K$, thus the total MLLR adaptation time is $\tau \times \sum_{k=1}^{P_s'} |s^{(k)}| = \tau \times K$, as in the baseline sequential case.

(c) *Parallel, with regression classes*:

In a parallel processing environment, we assume that we have Π processors that can work in parallel (for example, in a cluster computer[4], or in a multi-core architecture). In this case, Π adaptation processes can be performed in parallel. Assuming that, for an adaptation stream s, we need to perform $P_s' \leq P$ adaptations, two cases arise:

1. $\Pi \geq P_s'$: In this situation, each adaptation (using $|s^{(k)}|$ acoustic frames, for $k = 1, ..., P_s'$) can be performed in parallel, hence the computing time is given by $\tau \times \max_{k \in \{1,...,P_s'\}} |s^{(k)}| < \tau \times K$. The latter inequality always holds if $P_s' > 1$, that is, we have more than one phoneme acoustically realized in adaptation stream s. This is *usually* (but not always!) true in spoken dialogue (where we can have speech turns composed of interjections, e.g. "aaa!"). In those (relatively rare) cases where $P_s' = 1$, obviously, the parallelization strategy described here does not bring any performance improvement, concerning the computation time.
2. $\Pi < P_s'$: In this situation, only Π MLLR adaptation processes can be performed in parallel, at a time. Denoting by ρ the "process-to-processor

[4] We should however note that in a cluster computer the total processing time might be slightly higher, due to interprocessor communication delays.

ratio", $\rho = \left[\frac{P'_s}{\Pi}\right]$, where $[x]$ denotes the integer part of x, we have a *sequence* of ρ sets of *parallel* Π adaptations; thus, the total adaptation time is lower than $\tau \times (\sum_{j=0}^{\rho-1} \max_{k \in \{j \cdot \Pi+1,\ldots,(j+1)\cdot\Pi\}} |s^{(k)}| + \max_{k \in \{\rho\cdot\Pi+1,\ldots,P'_s\}} |s^{(k)}|) < \tau \times K$. The total computing time is given by the latter estimation if each set of Π adaptation processes have to finish completely before a new set of adaptations start. Obviously, this is neither optimal, nor compulsory; indeed, the adaptation processes can be performed such that as soon as one transformation matrix computation has finished in a set of Π parallel adaptations, a new MLLR process can start. In this case, of course, there is an upper bound of the computation time on average lower than that given by the maximum operator on $|s^{(k)}|$.

Obviously, out of these two cases, the second one is more interesting for practical purposes, if we assume that a phoneset in a language averages 35 ($P \approx 35$), and if the actual number of phone types acoustically realized in a data stream s averages 15 ($P'_s \approx 15$), and that in a (relatively widespread) dual-core computer (or four-core server) we have a relatively low number of processors ($\Pi \in \{2; 4\}$).

Finally, we should remark that while the baseline speaker tracking strategy was unsupervised, since adaptation data did not need to be decoded before performing MLLR adaptation, the parallel MLLR algorithm needs the word level (and, hence, phoneme-level) transcription of adaptation data. Therefore, parallel MLLR is supervised, but this does not add any further computational cost, since every input utterance is decoded using the speaker independent HMM set, before any adaptation process takes place. Moreover, the usage of phoneme-level regression classes brings further improvements, in both recognition rate and speaker tracking performance, as shown in studies like [23] or [13].

4 Experiments

4.1 Speech Recognition System

The algorithm described in this chapter was applied in a continuous speech recognition system, designed for "virtual librarian" multi-party dialogue applications, in Romanian language. The system was trained at word level, using no language modelling information. Thus, 92 words, related to library services were used, along with a supplementary set of 16 cue words; for each word a left-right (Bakis) [6] HMM was trained, with a variable number of states for each word (equal to 2 (initial and final, non-emissive states) + 3 $\times$ the number of phonemes in the word); the output observations are modelled with one Gaussian for each emissive state.

Each word-level HMM was trained in a speaker-independent manner, using around 4 hours of recorded speech (in laboratory conditions: SNR $\geq$ 25 dB), containing these words, uttered in context. The acoustic characteristics of the training data are (i) acquisition: unidirectional head-set microphone; (ii) sampling frequency: 16 kHz; (iii) signal frame size: 300 samples; (iv) weighting window type: Hamming; (v) parameterization: 12 MFCC (mel frequency cepstrum coefficients) per frame, along with the energy and with the first and second-order derivatives of these features; this results in a total of 39 acoustic features per frame.

The training was performed in two steps, following a classical "isolated unit training" strategy, based on hand-labeled data (at a word level)[5]:

1. the parameters of the set of *prototype* word-level HMMs[6] are initialised through Viterbi alignment to the data [22];
2. the parameters of the initialised HMMs are re-estimated via the Baum-Welch procedure [6].

The system achieves a word-level accuracy of around 79 % when tested against spontaneously-uttered speech produced (in laboratory conditions, akin to those used in training) by locutors not used in the training process; this relative low percentage can be explained through the spontaneous nature of the test utterances, where the word boundaries are not easy to localise and, moreover, words exhibit incomplete or altered acoustic realisation.

We stress on the fact that language modelling was not used in the system; we did not even use phonemic acoustic modelling, since the speech recognition system can be and is used as a word spotter for dialogue purposes: bearing in mind that the ultimate goal of speech recognition in dialogue contexts is to *analyse* the utterance from a semantic and pragmatic point of view [3], spotting words that are relevant to the task (that convey semantic information) and cue words (that convey discourse and pragmatic information) is more effective and efficient that full-blown continuous speech recognition [11], [10]. However, in the experiments described in this chapter, the utterances (for training and testing) were chosen so that they contain only the words considered, and the system was used as a speech recognizer.

4.2 Baseline Speaker Tracking Performance

The speaker tracking procedure was tested using a maximum of four speakers in multi-party dialogues. The dialogues are driven by a set of scenarios involving several typical tasks: (i) enrollment of a new customer of the library; (ii) request for books on a certain topic or subject; (iii) request for a specific book; (iv) restitution of a book; (v) payment of fines due to delays in book restitution.

[5] The labels include temporal information as well.

[6] The prototype HMMs are specified by the user and contain constraints on the size and topology of the models.

Table 3. Runtime variations in various configurations.

$\Delta \diagdown \overline{L}$	1	2	3	4
1	0.6674	0.7855	0.8843	0.9638
2	N/A	0.7975	0.8963	0.9759
3	N/A	N/A	0.9084	0.9879
4	N/A	N/A	N/A	1.0

The multi-party dialogues are constructed so that every possible speaker order and "weight" (in terms of number of turns per speaker / number of turns / conversation) is achieved; for the moment, given the fact that the entire dialogue system is, for the time being, a work in progress [3], the conversations are only between humans, out of which one plays the part of the librarian. Thus, a number of around 400 conversations were used for testing the ability of the algorithm to map utterances to the appropriate speaker identifiers.

It is worth mentioning that the number of speakers (between two and four, the maximum, as stated above) and their utterances were not previously known to the system; moreover, the speakers used in testing were different from those used in training the speaker-independent speech recognizer described in §4.1.

The first relevant performance figure in our experiment is given by evaluating the word recognition of a system that is adapted to a certain speaker via unsupervised MLLR, on an utterance of that speaker, versus the same measure obtained, in the same test conditions, with the speaker-independent system. Thus, the average word recognition rate on a set of utterances produced by a certain speaker (out of those used in testing) reaches more than 80 % for the adapted system, versus around 79 % for the speaker-independent system.

The most relevant performance measure of our algorithm is represented by the number of correct speaker identifier assignments for each utterance, divided by the total number of utterances (since each utterance has a speaker identifier associated with it). Moreover, we studied the variation of this ratio (denoted by ρ in this chapter) with respect to the parameters $\overline{L}$ and Δ, the "expected" number of speakers and the offset, respectively (see §2 for details). Thus, $\overline{L}$ was varied from 1 to 4, and Δ, from 1 to $\overline{L}$ (see §2.2 for the rationale behind the choice of Δ). A score of 81.2 % was obtained for ρ; this score is independent of $\overline{L}$ and Δ, which makes sense, since these parameters control the strategy adopted for speaker tracking in respect with runtime, rather than with the actual decisions being made. Thus, a more interesting evaluation concerning these parameters is related to the runtime of the algorithm.

However, since in our experiments the algorithm was implemented as a series of Bash and Python scripts driving HTK (Hidden Markov Modelling Toolkit) tools [22], absolute runtime values are not relevant, preferring relative values instead, scaling them with respect to the runtime for $\Delta = \overline{L} = 4$; thus, we mark by 1.0 the runtime of the algorithm for the values specified just above

Table 4. Confusion matrix for speaker identity assignments.

	M_1	F_1	M_2	M_3
M_1	**1296**	8	112	184
F_1	88	**1408**	8	96
M_2	180	12	**1216**	192
M_3	144	4	172	**1280**

for $\overline{L}$ and Δ (hence, this is the baseline case), and providing relative coefficients for the other combinations for these two parameters; details are given in Table 1, for a series of around 400 dialogues, each one counting between 4 and 20 utterances (speech turns).

From Table 3 we see that the fastest speaker tracking algorithm is actually obtained for the minimal values for $\overline{L}$ and Δ and the runtime increases with both these parameters; this proves that, after only the first utterance had been processed (i.e., a first speaker identity assigned to it, and the speech recognition system adapted to it), it is worth adopting the "lazy" speaker tracking strategy stated in steps 5. and 6. in the algorithm. That is, the algorithm could be simplified, in that steps 3. and 4. could be eliminated completely, running directly steps 5. and 6., from the second input utterance on (i.e., after step 2.); this is more efficient because in these latter steps first testing is performed, and then, if necessary, further adaptation, whereas in steps 3. and 4., adaptation is performed first, then testing and, if necessary, adapted systems are discarded. However, for the first input utterance, it makes sense to perform first the adaptation, as in step 2.

Yet, a further refined analysis of the performances of the algorithm can be emphasized, namely by showing the confusion matrix concerning the assignment of speaker identities to utterances. Thus, denoting by M_1, M_2 and M_3 the three male speakers, and by F_1 the female speaker used in testing the algorithm, the confusion matrix is shown in Table 4, where the figures on the lines indicate the assignments performed by the algorithm.

From this confusion matrix, *precision* (defined as the number of correct speaker assignments, divided by the total number of speaker assignments) and *recall* (defined as the number of correct speaker assignments, divided by the total number of real speaker-to-utterance mappings) can be derived, for each speaker and, consequently, the corresponding F-measures (defined as the harmonic mean of precision and recall). These quantities are shown in Table 5 for each of the four test speakers considered. We can see that these quantities are evenly balanced, although, as we could expect, the best results are obtained for the (only) female speaker. This shows, on the one hand, that the performances of the algorithm are robust to speaker variations, and that the usage of only one female speaker introduces an artificial bias on the results. Therefore, the most relevant performance figures are those obtained for the male speakers. Moreover, we can see that there is a balance between precision

Table 5. Performance measures for every test speaker.

	Precision	Recall	F-measure
M_1	0.76	0.81	0.78
F_1	0.98	0.88	0.93
M_2	0.81	0.76	0.78
M_3	0.73	0.80	0.76

and recall as well, which might be a hint that further tests are needed in order to see whether this is a feature of the algorithm, or of the test data.

Concerning the values of the recognition scores obtained, the word-level recognition log-likelihood scores evolve around -2000 for adapted systems recognizing utterances produced by the speakers that the systems are adapted to, and around -3700 for the non-adapted speaker-independent system. On the other hand, the word-level log-likelihood score variances are in the range of around ± 800, thus, less than the difference between the first two average scores. Therefore, tests indicate that the usage of log-likelihood scores is a reliable strategy for speaker tracking.

4.3 Performance Improvements

The performance optimizations that stem at parallelizing the speaker tracking algorithm yield manyfold effects: first, a reduction in the processing time is achieved (considering both parallel recognition of input utterances, with speaker independent and speaker adapted HMM sets, and parallel MLLR). Secondly, when multiple regression classes are considered, speaker tracking and speech recognition accuracies improve. In this section one will show some quantitative effects of the parallel speaker tracking algorithms.

First, runtime improvements regarding the speaker tracking algorithm where speech decoding processes run in parallel, versus the case where they run in sequence have already been discussed in previous research [23], [13], hence here we only say that, given the fact that these recognition processes bare a significant (average) weight (of around 75 %) in the total tracking time, for a given utterance, the performance gain is mostly important as the number of speakers increases, and the number of processing units available is at least equal to the the number of speakers.

Secondly, word error rate improvements of around 0.5 - 1.5 % have been reported in previous studies [23], for the case where phoneme-level regression classes are used in MLLR adaptation, versus the case where no regression class is used (i.e., only one transformation matrix is computed for all the states in the HMMs). Hence, in this chapter we will emphasize the less studied issue of the effects of MLLR parallelization on the total speaker tracking runtime, for each new input utterance. Thus, performance can be characterized by looking at two measures: first, the ratio between the time taken by MLLR adaptations, and the total speaker tracking time, for an input utterance; we denote this

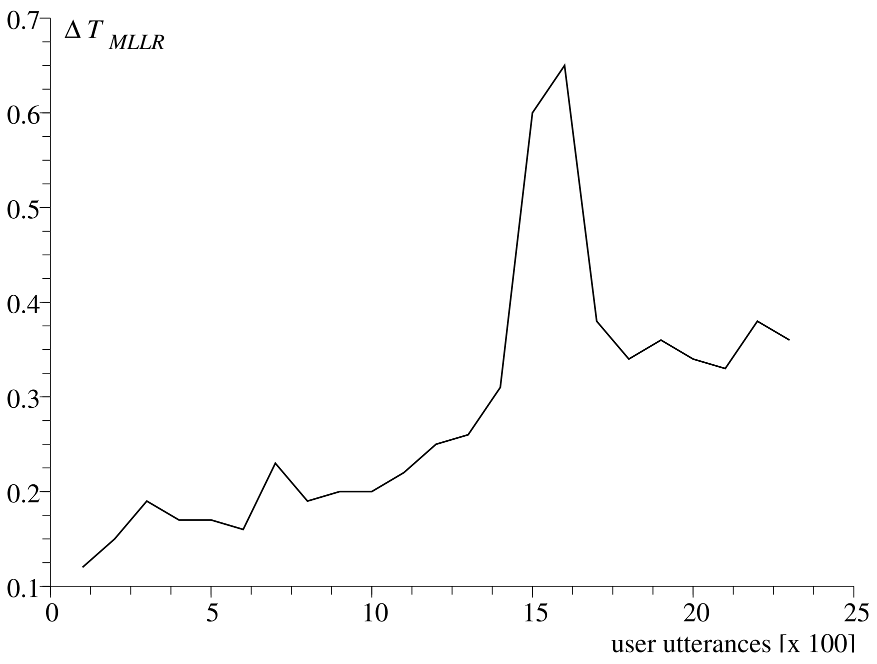

Fig. 2. Weight of MLLR time in a speaker tracking process.

measure by ΔT_{MLLR}. Then, the ratio between the time taken by a parallelized MLLR process, and a sequential (normal) MLLR adaptation can be considered; we denote this measure by $\Delta T_{\|MLLR}$. Obviously, $\Delta T_{MLLR} \in [0;1)$, since we can have situations where no adaptation is performed for a tracking process, and $\Delta t_{\|MLLR} \in (0;1]$, since there are situations where parallel MLLR is not faster than sequential MLLR (e.g., for utterances composed of a sequence repeating the same phoneme type).

For the beginning of a dialogue, where speakers begin to take part in conversation, MLLR adaptation is more important (since, depending on the values of $\overline{L}$ and Δ (see §4.2), for the utterances produced until the number of speakers stabilizes, MLLR adaptations take place roughly for any speaker tracking process). Afterwards, MLLR adaptations only take place if new speakers occur in dialogue, or if there are at least two speakers with very similar voices (therefore, supplementary adaptation processes are needed). As for the advantages that parallel MLLR adaptation brings, with respect to sequential MLLR, they are more perspicuous if several phoneme types are instantiated in input utterances (according to decoders' outputs). On the other hand, if too many phoneme types are instantiated in an utterance, word error rate can degrade, due to insufficient adaptation data.

The variation of ΔT_{MLLR} is shown in Figure 2, in the context of speech recognition being ran in parallel, over $\Pi = 4$ processors; hence, the figure

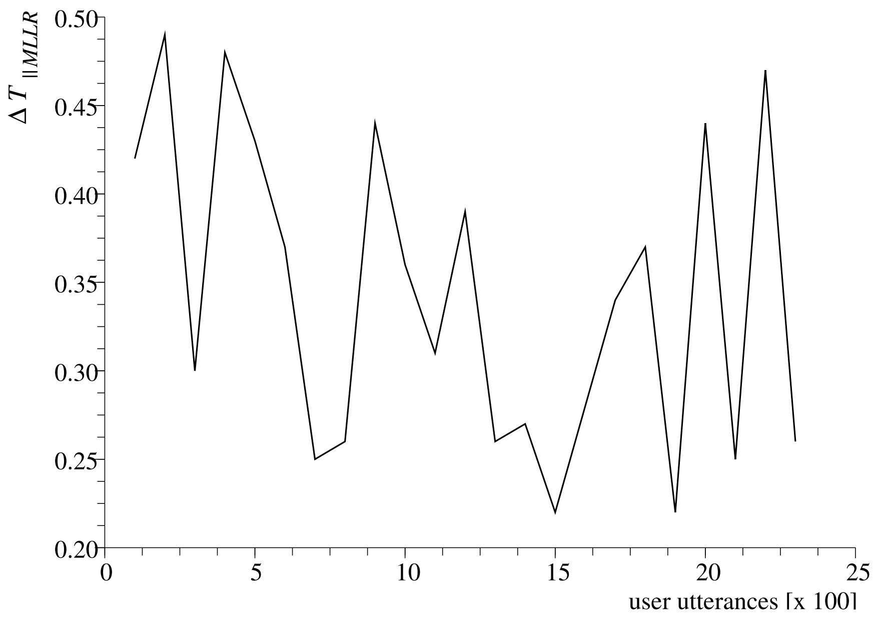

Fig. 3. Parallel MLLR runtime performance, for $\Pi = 4$ processors.

shows the weight of the (sequential) adaptation time in the total tracking time, for an utterances. On the horizontal axis user utterances (scaled by a factor of 100) are plotted, for the four users in the 400 dialogues considered in §4.2; on the vertical axis, ΔT_{MLLR} is plotted. We can see that, on average, MLLR takes around 30 % of the total speaker tracking time, for an input utterance.

As for $\Delta T_{\|MLLR}$, that quantifies the runtime effects of parallelization on MLLR alone, it is shown in Figure 3; the same conventions and context as in Figure 2 are used. We can see that, when $\Pi = 4$ processing units are used in parallel, runtime reductions of more than 50 % are achieved. Theoretically, $\Delta T_{\|MLLR}$ should be around 25 %; however, due to uneven phoneme balance in utterances, and to inherent operating system-determined computational load, variable percentages are obtained.

The two figures, 2 and 3 altogether show that via MLLR parallelization an overall performance improvement in speaker tracking runtime of $\Delta T_{MLLR} \times (1 - \Delta T_{\|MLLR})$ is obtained, that is, on average, around 19 %, for each input utterance.

5 Conclusions

We have presented a computationally simple strategy for speaker tracking in multi-party human-computer dialogue. The approach is based on the traversal of a decision tree that relies on speech recognition scores and unsupervised MLLR adaptation of a speech recognition system to input utterance so that these scores are maximized. Thus, an algorithm linear in the number of previous utterances in dialogue is obtained, that achieves a speaker tracking performance of around 80 % on spontaneous speech. The algorithm has been tested in the context of a virtual librarian dialogue application, in Romanian and French languages, and exhibits good runtime performance.

Moreover, several performance improvements were proposed, especially concerning the runtime of the speaker tracking algorithm; they basically rely on the idea of parallelizing the adaptation process. We have studied a parallel MLLR adaptation strategy that relies on simultaneous adaptation of HMM states grouped in regression classes, whereby MLLR runtime is reduced by a factor proportional to the number of processors available. Thus, taking into account the weight of roughly 25 - 30 % that MLLR adaptation has in the total speaker tracking time, for an input utterance, we remark that for a common four-core processor, total speaker tracking time is reduced by a factor of about 19 %, on average.

However, several issues remain to be done for tuning a speech recognizer into a front-end to multi-party dialogue systems; for instance, the algorithms proposed in this paper could be improved with contextual information (expressed at semantic, or even *discourse* levels [3]): a certain speaker is more likely to say certain things, or, viceversa, certain things were more likely said by a certain speaker. Thus, a statistical model of user conversational preferences could be trained, starting from discourse structures (rhetorical relations between utterances or sets of utterances) where the user's utterances integrate.

Another rather important technical detail that might be improved concerns the way we built the regression classes for the MLLR adaptation: we either did not use any regression class at all (by computing only one, global, transformation matrix for all the states in the HMM set), or we used phoneme-level regression classes (where for each phoneme type we compute a transformation matrix for all the states that correspond to that phoneme type). Neither of these two approaches is optimal, in the sense that the former is too coarse (which impends on the reliability of the transforms computed), whereas the latter is too fine-grained (which poses problems in case of data sparseness). Hence, a better approach would be to follow a strategy similar to that implemented in the HTK toolkit [22]: MLLR could use a regression class tree to group Gaussians in the HMM set, so that the set of transformation matrices to be computed can be chosen according to the amount and type (e.g. phonemes contained) of the adaptation data available. Thus, the tying of each transformation matrix across several mixture components makes it possible to adapt distributions for which no adaptation data was available. In our current setting, such distributions are not adapted unless relevant data (i.e., that contains acoustic realizations of the phoneme type that indicates the regression class referred) is available.

References

1. Barras C (1996) Reconnaissance de la parole continue : adaptation au locuteur et contrôle temporel dans les modèles de Markov cachés., PHD Thesis, University of Paris VI, Paris
2. Braningan H (2006) Research on Language and Computation 4:153–177
3. Caelen J, Xuereb A (2007), Interaction et pragmatique - jeux de dialogue et de langage. Hermès Science, Paris
4. Christensen H (1996) Speaker adaptation of hidden Markov models using maximum likelihood linear regression. MA Thesis, University of Aalborg, Denmark
5. Ginzburg J, Fernandez R (2005) From Dialogue to Multilogue... In: Proc. of ACL
6. Huang X, Acero A, Hon H-W (2001) Spoken language processing: a guide to theory, algorithm and system development. Prentice Hall, New Jersey
7. Landragin F (2005) Dialogue homme-machine multimodal. Hermès Science, Paris
8. Larsson S, Traum D (2000) Natural Language Engineering 1(1): –
9. Leggetter C J, Woodland P C (1995) Computer Speech and Language 9:171–185
10. McTear M F (2002) ACM Computing Surveys 34(1):90–169
11. Minker W, Bennacef S (2001) Parole et dialogue homme-machine. CNRS Editions, Paris
12. Motlicek P, Burget L, Cernoký J (2005) Non-parametric speaker turn segmentation of meeting data. In: Proc. Eurospeech, Lisbon
13. Murani N, Kobayashi T (2003) Systems and Computers in Japan 34(13):103–111
14. Popescu V, Burileanu C (2005) Parallel implementation of acoustic training procedures for continuous speech recognition. In: Burileanu C (ed) Trends in speech technology. Romanian Academy Publishing House, Bucharest
15. Popescu V, Burileanu C, Rafaila M, Calimanescu R (2006) Parallel training algorithms for continuous speech recognition, implemented in a message passing framework. In: Proc. Eusipco, Florence
16. Popescu-Belis A, Zufferey S (2007) Contrasting the Automatic Identification of Two Discourse Markers in Multi-Party Dialogues. In: Proc. of SigDial, Antwerp
17. Ravishankhar M (1996) Efficient algorithms for speech recognition. PHD thesis, Carnegie Mellon University, Pittsburg
18. Sato S, Segi H, Onoe K, Miyasaka E, Isono H, Imai T, Ando A (2004) Electronics and Communications in Japan 88(2):41–51
19. Trudgill P, Sociolinguistics: an introduction to language and society. (fourth edition) Penguin Books, London
20. Yamada M, Baba A, Yoshizawa S, Mera Y, Lee A, Saruwatari H, Shikano K (2005) Electronics and Communications in Japan 89(3):48–58
21. Yamada S, Baba A, Yoshizawa S, Lee A, Saruwatari H, Shikano K (2005) Electronics and Communications in Japan 88(8):30–41
22. Young S, Evermann G, Kershaw D, Moore G, Odell J, Ollason D, Povey D, Valtchev V, Woodland P (2005), The HTK book. Cambridge University, United Kingdom
23. Zhang Z, Furui S, Ohtsuki K (2002), On-line incremental speaker adaptation for broadcast news transcription, Speech Communication 37:271–281