



A Comparison of Speaker Clustering and Speech Recognition Techniques for Air Situational Awareness

Wade Shen and Douglas Reynolds

MIT/Lincoln Laboratory
 244 Wood St.
 Lexington, MA USA 02420
 {swade, dar}@ll.mit.edu

Abstract

In this paper we compare speaker clustering and speech recognition techniques to the problem of understanding patterns of air traffic control communications. For a given radio transmission, our goal is to identify the talker and to whom he/she is speaking. This information, in combination with knowledge of the the roles (i.e. takeoff, approach, hand-off, taxi, etc.) of different radio frequencies within an air traffic control region could allow tracking of pilots through various stages of flight, thus providing the potential to monitor the airspace in great detail. Both techniques must contend with degraded audio channels and significant non-native accents. We report results from experiments using the nn-MATC database [6] showing 9.3% and 32.6% clustering error for speaker clustering and ASR methods respectively.

Index Terms: speaker clustering, robust speech recognition, entity detection

1. Introduction

Basic communication patterns in air traffic control (ATC) speech provide key information about the pilot movements and activities. As pilots operate within an air traffic control region, they communicate with different controllers requesting different activities (e.g. landing, takeoff, etc.). As a result, an understanding of the speaker communication patterns can help to track pilot activities within an ATC region. This information can be further supplemented by understanding of individual utterances spoken between controllers and pilots using language processing techniques [7].

While typical communications between pilots and controllers follows strict protocols, allowing for constrained pattern processing (e.g., grammars or turn taking), the acoustic signals are typically short and over noisy and corrupted channels adding some challenges for speech processing algorithms.

In this paper we examine and compare two methods of automatic extraction of speaker information from ATC communications. First, we describe experiments using speaker clustering and speaker recognition techniques to 1) decide whether a transmission came from a pilot or controller and 2) group different transmissions from a single pilot together. We compare this technique with ASR methods [5] in which each pilot/controller transmission is transcribed by ASR after which

callsigns are identified using named-entity recognition and parsing techniques. Pilot/controller transmissions are then grouped by callsigns.

In both cases, we derive an association between utterances and speakers assessed by the clustering error rate:

$$E_C = \frac{N_{error}}{N_{seg}}, \quad (1)$$

This measure is simply the segment-weighted counterpart of the NIST speaker diarization error measure [4].

The paper is organized as follows: section 2 describes the speaker clustering and speaker recognition techniques we applied to this problem. In section 3 we describe the ASR system that we developed to handle corrupted ATC speech, and section 4 details the natural language processing techniques we employed for callsign identification. Section 5 compares clustering results from these methods individually and in combination.

2. Speaker Clustering

Audio diarization, the process of annotating an input audio channel with information that attributes (possibly overlapping) temporal regions of signal energy to their specific sources, typically consists of several steps [2]:

1. Speech activity detection to find regions of speech in the audio.
2. Change detection to find speaker change points in speech regions
3. Clustering of speech segments coming from the same speaker (possibly on automatically labeled subsets (e.g., male/female).
4. Cluster recombination
5. Re-segmentation using putative speaker clusters

For the ATC task in this paper, it is assumed that speech segments (transmissions) are automatically detected and that each transmission contains a single talker. Thus the focus here is only on the clustering stage of the diarization process.

We use a hierarchical, agglomerative clustering system with a BIC based stopping criterion [1] consisting of the following steps:

0. Initialize leaf clusters of tree with speech segments.
1. Compute pair-wise distances between each cluster.
2. Merge closest clusters.

This work was sponsored by the United States Air Force Research Laboratory under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

3. Update distances of remaining clusters to new cluster.
4. Iterate steps 1-3 until stopping criterion is met.

The clusters are represented by a single full covariance Gaussian and a generalized likelihood ratio (GLR) with a BIC penalty term is used as the distance metric. This distance compares the penalized likelihood between keeping two clusters, x and y , separate versus merging them into a single cluster, z , and is computed as

$$\Delta\text{BIC} = \frac{1}{2} [N_z \log(|S_z|) - N_x \log(|S_x|) - N_y \log(|S_y|)] - \alpha P$$

$$P = \frac{1}{2} \left(d + \frac{d(d+1)}{2} \right) \log(N) \quad (2)$$

where N is the number of feature vectors, S is the covariance matrix and d the dimension of the feature vector.

For each step, the pair of clusters with the lowest ΔBIC is merged and the statistics are recalculated. The process is generally stopped when the lowest ΔBIC is greater than a specified threshold, usually 0. The BIC weight, α is set to a value around 1.0 (1.2 in this work).

In the experiments we used two different feature sets for pilot-controller clustering and for pilot-only clustering. Since bandwidth is a big differentiator between pilot and controller communications, we simply used 4 cepstral coefficients derived from a full (8kHz) bandwidth filterbank. These low order cepstra are basically modeling a highly smoothed representation of the channel passband. For pilot-only clustering, we needed more spectral detail, so used 25 cepstra coefficients from the full bandwidth. No form of channel compensation were used since the channel (cockpit noise and radio channel) is correlated with the speaker and aids in the clustering process.

3. Speech Recognition of Non-native ATC Speech

For this task, we used a standard HMM-based recognizer trained on the nn-MATC training set (18.36h). As the data was quite noisy and the channels quite varied, we experimented with a number of front-end and model-space compensation techniques (shown in Table 1). Detailed results are described in Section 5. Speaker-independent, state-clustered triphone models

Table 1: Front End and Model Compensations Employed

Front End	Model Space
RASTA	MLLR (mean-only)
MMSE-LSA Noise Reduction	Constrained MLLR
Feature Gaussianization	
HLDA	

were trained using data from both pilots and controllers with 8 diagonal covariance Gaussian components per state. Following speaker-independent training, SI models were used to estimate one global HLDA transform using the EM procedure described in [11]. Then speaker adaptive training (SAT) using CMLLR transforms was performed using entire audio files (generally corresponding to single controllers, but including multiple pilots) for transform training. Adaptive training on a per file basis improves performance only slightly ($\approx 0.5\%$ in most experiments).

We experimented with models trained separately for controllers and pilots, but this resulted in higher overall word error

rates. In doing these experiments, we found a high discrepancy in the performance of our ASR system on these two data sets (in optimal configuration, 37.4% WER (controller) and 69% WER (pilot)). This could be due to a wider variety of accent and channel differences amongst pilots.

4. Call Sign Identification

We used a hidden n-gram/tagger-based statistical system to extract flight ID information from transcripts generated by the ASR system. Our system (FLID) uses an integrated trigram language model with interpolated Knesser-Ney smoothing [8]. Additionally, a history model is used to identify potentially abbreviated IDs that can be missed by the underlying statistical model. Output from this system is shown in figure 1.

Figure 1: Example Callign ID Output

do you have intentions of the [tiger one two] [beep]
[belgian air force four three two] continue descent maintain heading

4.1. Tagging Model

We treat callsign identification as an HMM-based tagging in which callsign *tags* are unobservables states of a Markov process. In this framework, words are assumed to be generated by the underlying state sequence.

The basic model can be described as follows. Let

$$\mathbf{w} \equiv w_1 \dots w_k \quad (3)$$

$$\mathbf{s} \equiv s_1 \dots s_k \quad (4)$$

Where \mathbf{w} is a sequence of observed words of length k and \mathbf{s} is a sequence of hidden states (tags).

$$P(\mathbf{w}|\mathbf{s}) = P(s_1)P(w_1) \prod_{i=2}^k P(w_i|s_i)P(s_i|s_{i-1})$$

$$P(\mathbf{w}) = \sum_{\forall \mathbf{s}} P(\mathbf{w}|\mathbf{s}) \quad (5)$$

Each word is generated by exactly one state in \mathbf{s} . For the purposes of identifying callsigns given a specific word string as input, our goal is to find the optimal state sequence $\hat{\mathbf{s}}$ such that

$$\hat{\mathbf{s}} = \arg \max_{\mathbf{s}} P(\mathbf{w}|\mathbf{s}) \quad (6)$$

In this basic model, states represent the status of each word (see Table 2). This is similar to methods used for shallow bracketing and base-NP tagging [9] [10].

Table 2: States of Callsign ID HMM Model

[beginning of callsign
]	end of callsign
<1>	callsign internal word
<0>	and non-callsign word

The callsigns in this corpus are quite structured when they are not abbreviated. They typically begin with a closed-class, head word (belgian or tiger) followed by a sequence of

numbers or military letters. We also note that the length of callsigns is often predictable based on the callsign head word. Because of the structured nature of their unabbreviated forms, we model callsigns using word class information (e.g. `number` vs. `head-word` vs. `letter`) in addition to duration.

We extend this basic bracketing state structure by incorporating word and word class context information in addition to a duration model as follows.

$$s_i = \langle w_{i-n-1} \dots w_i, c_{i-n-1} \dots c_i, dur_i, sb_i \rangle \quad (7)$$

where n is the length of the context history c_i is a class associated word i , dur_i is the length of the current callsign (or zero if s_i is not callsign internal), and sb_i is the basic bracket tag (see Table 2).

This structure allows for dependency between prior states, words and word classes for output and transition probabilities. We use a simple transition structure:

$$P(s_i | s_{i-1}) \approx P(wc_i, sb_i | wc_{i-n-1} \dots wc_{i-1}) * P(sb_i | head(w_{i-1}), len_{i-1}) \quad (8)$$

where $head(w_{i-1})$ is the first word of the current callsign if s_{i-1} is callsign internal, or NULL if s_{i-1} is callsign external. The corresponding output probabilities are modeled as:

$$P(w_i | s_i) \approx P(w_i | w_{i-n-1}, sb_i) * P(s_i | s_{i-1}) \quad (9)$$

4.2. History Model

In many cases callsigns can be abbreviated into a sequence of digits. The abbreviated form is typically a callsign from prior transmissions. In addition to the segment-independent callsign identification model described above, we maintain a queue history of current callsigns. As new utterances are processed and callsigns detected, new flight IDs are pushed into the queue as older callsigns are removed. For experiments conducted in this paper, a queue of the last four unique callsigns was maintained. When the callsign ID system hypothesizes no callsigns for a given utterance, substrings of each callsign (greater than length 3) in the history queue is matched against the current utterance by expanding queued callsigns into pronunciation phone sequences and computing the string edit distance between the current callsign and a substring of the utterance (that has also been expanded by the pronunciation dictionary). Matching strings are then used to hypothesize abbreviated callsigns. In prior experiments, this simple procedure improved callsign identification recall by 14% relative without significant impact on precision (2.6% relative) on a similar civilian callsign ID task.

4.3. Speaker Clustering by Callsign

In addition to detecting callsigns that were missed, we used the pronunciation matching facility in our history model to group alternate versions (i.e. those that are abbreviated or corrupted by ASR errors) of a callsign into clusters. For speaker clustering, each utterance is associated with a callsign cluster (typically one exists per utterance).

5. Results

We ran experiments using the `nn-MATC` database using the defined test set to compare the performance of speaker clustering and ASR-based callsign detection techniques. The 18.36 hour training partition was used to train both the ASR and callsign ID systems. For both systems, we performed two clustering tasks on a per segment basis using the test set:

1. pilot/controller clustering (2 clusters)
2. pilot speaker clustering (23 pilot clusters)

We evaluated the performance of both tasks using the clustering error rate as the basic metric. The error rate was computed using the NIST `md-eval-v19.pl` tool [4].

We used only the acoustic speaker clustering system for task 1 as callsigns are generally not distinguishing for controllers. Task 2 assumes that pilot/controller information has been provided upfront. Thus, the performance of our system on this task is optimistic relative to operation in a real application.

The cascade of tasks 1 and 2 most closely mimics real-world operating conditions as no information other than segmentation is assumed upfront. As the database has already been automatically segmented [6], it seems reasonable to assume that segmentation could be achieved using existing techniques. By performing task 1 followed by task 2, errors made during pilot/controller clustering cannot be recovered from. Subsequent pilot speaker clustering will incur false alarms or misses corresponding to every error made upstream.

To preprocess data for the aforementioned task we used pilot speaker information provided in the database (based on reference callsigns), and manually normalized and combined abbreviated versions of callsigns with their unabbreviated forms to generate larger clusters in the reference (which are presumably better represent speakers). The resulting reference contained 86 pilot and 159 controller transmissions for task 1 and 23 unique pilot clusters across the 86 pilot transmissions for task 2. To eliminate extraneous silence, we automatically preprocessed each utterance by removing leading and trailing silences. This was done by performing ASR decoding with forced silence at the beginnings and endings of utterances.

5.1. ASR Performance

We compared a number of configurations of our ASR engine with the results shown in Table 3. Both MMSE-LSA (Mini-

Table 3: ASR Configurations

Front End	Noise Compensation	LM order	WER
PLP	CMN/CVN	bigram	50.5%
MFCC	CMN/CVN	bigram	49.3%
MFCC	MMSE-LSA	bigram	49.1%
MFCC	RASTA	bigram	48.5%
MFCC	Gaussianization	bigram	47.8%
MFCC	Gaussianization	trigram	45.5%

mum Mean Square Error – Log Spectral Amplitude) [12] and RASTA [13] improve the performance of our ASR system only slightly, perhaps because of the relatively short duration of the transmissions and the minimal amount of silence per utterance. Feature-space Gaussianization [14] provided a small improvement over both techniques. The final ASR configuration used for ASR-based clustering is shown in bold.

5.2. Callsign Performance

The callsign identification system described in section 4 was applied to the pilot clustering task. Since the error rate of pilots is significantly higher than that of controllers (67% vs 33%), for callsign identification experiments we used *both* pilot and controller utterances to assist the history model. This allows for partially corrupted (high WER) or missing callsigns from a

pilot utterance to be associated with a lower WER version in a controller utterance.

Table 4 shows the performance of the callsign identification system with and without the history model. Here precision, recall and accuracy are computed at the word level treating callsign identification as a tagging problem in which each word is either a callsign word or not. This gives “partial credit” for partially matched callsigns. These measures can be contrasted with the final column showing the rate of complete callsign matches produced by the system. These results are significantly worse

Table 4: Callsign Identification Results

Configuration	Tagging Performance			Match Rate
	Precision	Recall	Accuracy	
Basic Model	94.98%	77.17%	94.66%	82.26%
+ History	93.76%	85.50%	95.99%	85.48%

than results obtained using the DCA set of the ATCO corpus from LDC [15]. The same system parameters applied to that task give a complete callsign match rate of 94.68%.

5.3. Pilot/Controller Clustering Performance

For the pilot/controller clustering task only the acoustic speaker clustering system was applied. Using 4 cepstral coefficients and forcing 2 cluster outputs, we obtain $E_C = \frac{1}{245} = 0.4\%$. As expected from previous work [3], the bandwidth and channel differences between pilots and controllers make this a rather easy task.

5.4. Pilot Speaker Clustering Performance

In Table 5 we show the results for pilot speaker clustering for the acoustic speaker clustering system and the callsign-based clustering system (with reference and with ASR transcripts). The acoustic speaker clustering system obtains a relatively low error rate for the more challenging pilot transmissions. The system produced 23 clusters, but these were not in 1:1 agreement with the unique pilots. With reference transcripts the callsign-based clustering has a about twice the error rate. With ASR transcripts the error rate again nearly doubles. This may be partly expected since the callsign clustering system is required to extract, adjudicate and apply more content from the speech than the acoustic speaker clustering system.

Table 5: Pilot Speaker Clustering Results

System	E_C
Acoustic Speaker Clustering	9.3% (8/86)
Callsign-based Clustering (reference)	17.44% (15/86)
Callsign-based Clustering (ASR)	32.56% (28/86)

6. Discussion

In this paper we have presented systems and results for extracting callsigns and clustering transmissions using the nn-MATC database that can be used for air situational awareness. Using acoustic information only, we demonstrated that unsupervised clustering techniques can be quite effective for pilot/controller and pilot-only transmission clustering. This clustering has the distinct advantage of being able to exploit the correlation of noise and channel effects with particular speakers. ASR-based

approaches, on the other hand, need to recognize spoken callsigns regardless of the speaker, noise or channel and so are severely challenged by these variabilites. Standard ASR compensation techniques help to some extent, but more gains come from using downstream processing to take advantage of the structured nature of the communications. A natural follow on to this study is to combine the strengths of the acoustic clustering with the content information from the callsign recognition to associate callsigns to speaker clusters either via a post-processing merge (e.g., count callsigns in cluster segments) or integrated into the clustering process (e.g., use a string distance between segment callsigns as a bias to the acoustic match score).

7. References

- [1] S. S. Chen and P. S. Gopalakrishnam, “Speaker, Environment and Channel Change Detection and Clustering via the Bayesian Information Criterion,” in *Proc. 1998 DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, 1998, pp. 127–132.
- [2] S. Tranter, D. Reynolds, “An Overview of Automatic Speaker Diarisation Systems”, *IEEE Transactions on Audio, Speech and Language Processing*, October 2006.
- [3] H. Gish, M.-H. Siu, and R. Rohlicek “Segregation of speakers for speech recognition and speaker identification”, *ICASSP*, pp. 873-876, 1991.
- [4] J. G. Fiscus, N. Radde, J. S. Garofolo, A. Le, J. Ajot, C. Laprun, “The Rich Transcription 2005 Spring Meeting Recognition Evaluation,” *MLMI*, 2005.
- [5] L. Denenberg, H. Gish, M. Meteer, T. Miller, J. R. Rohlicek, W. Sadkin, M. Siu, “Gisting Conversational Speech in Real Time”, *ICASSP*, vol. II pp. 131–134, 1993.
- [6] S. Pigeon, W. Shen, and D. Leeuwen, “Design and Characterization of the Non-native Military Air Traffic Communications Database (nnMATC)”, submitted to *Inter-speech 2007*.
- [7] M. Meteer, J. R. Rohlicek, “Statistical Language Modeling Combining N-gram and Context-free Grammars”, *ICASSP*, vol. I pp. 37–40, 1993.
- [8] Stanley F. Chen and Joshua Goodman. “An empirical study of smoothing techniques for language modeling,” *Computer Speech and Language*, October, 1999.
- [9] F. Sha and F. Pereira, “Shallow parsing with conditional random fields,” In *Proceedings of the HLT-NAACL Annual Meeting*, 2003.
- [10] M. Collins, “Three generative, lexicalised models for statistical parsing,” In *Proceedings of the 35th Annual Meeting of ACL*, pages 16-23, 1997.
- [11] M J F Gales, “Semi-tied covariance matrices for hidden Markov models,” *IEEE Transactions Speech and Audio Processing*, vol. 7, pp. 272-281, 1999.
- [12] D. Malah, R.V. Cox and A.J. Accardi, “Tracking Speech-Presence Uncertainty to Improve Speech Enhancement in Mon-Stationary Noise Environments,” *ICASSP*, 1999.
- [13] H. Hermansky and N. Morgan, ”RASTA processing of speech”, *IEEE Trans. on Speech and Audio Proc.*, vol. 2, no. 4, pp. 578-589, Oct. 1994.
- [14] G. Saon, S. Dharanipragada and D. Povey, “Feature Space Gaussianization,” *ICASSP*, 2004.
- [15] J. J. Godfrey, “Air Traffic Control Complete,” *Linguistic Data Consortium*, Philadelphia, 1994.