Real-Time ASR from Meetings

Philip N. Garner[1], John Dines[1], Thomas Hain[2], Asmaa El Hannani[2], Martin Karafiát[3], Danil Korchagin[1], Mike Lincoln[4], Vincent Wan[2], Le Zhang[4]

[1]Idiap Research Institute, Martigny, Switzerland
[2]Speech and Hearing Research Group, The University of Sheffield, UK
[3]Speech Processing Group, Brno University of Technology, Czech Republic
[4]Centre for Speech Technology Research, The University of Edinburgh, UK

Phil.Garner@idiap.ch

Abstract

The AMI(DA) system is a meeting room speech recognition system that has been developed and evaluated in the context of the NIST Rich Text (RT) evaluations. Recently, the "Distant Access" requirements of the AMIDA project have necessitated that the system operate in real-time. Another more difficult requirement is that the system fit into a live meeting transcription scenario. We describe an infrastructure that has allowed the AMI(DA) system to evolve into one that fulfils these extra requirements. We emphasise the components that address the live and real-time aspects.

Index Terms: real-time speech recognition, meeting ASR, beam-forming, speech meta-data.

1. Introduction

AMI (Augmented Multi-party Interaction) was a European Union funded project aimed at analysing and aiding human-human interaction, specifically in the context of meetings. AMIDA (AMI with Distant Access) is the successor project. Whereas AMI focussed mainly on the concept of off-line meeting browsing, one of the main goals of AMIDA is to bring into AMI the concept of remote participants; that is, people not necessarily present in the same room where a meeting is taking place, but wishing to take part. The remote partipant could be present in a remote meeting room over a video conferencing link for the whole meeting, or could be present for some portion of the time, or via a device with limited capabilities. The remote participant concept brings with it the idea of on-line access, and hence real-time processing.

As well as remote participants, real-time meeting ASR (Automatic Speech Recognition) enables real-time augmentation. Although the accuracy of far-field ASR is not yet good enough for accurate real-time transcription, one persuasive application for which it is suitable is automatic content linking [1]. Words in the transcript are used to search for content relevant to the meeting, which is presented to the participants wherever their location.

In the context of ASR, AMIDA presents several challenges. Not least of these is the fact that the AMI system is multi-pass and off-line, operating in several times real-time. By contrast, the AMIDA requirement is for on-line and real-time or faster. Further, it involves sub-systems written in different (programming) languages by different people at different institutions.

2. The AMI system

The AMI system for meeting room recognition is a combination of beam-forming, diarisation and ASR. It has been the basis of several entries to the NIST RT evaluations [2]. In keeping with the RT evaluations, and the way data is collected in meeting rooms, the AMI system is actually two systems:

1. IHM (Individual Headset Microphone) is designed for close-talking microphones, one per participant. Whilst in practice there is crosstalk between the microphones, the basic assumption is that each channel carries the speech of just one speaker.
2. MDM (Multiple Distant Microphone) refers to a microphone array. ASR is done on the result of beam-forming, but without specifying the beam-forming algorithm.

The main practical difference between the two systems is that they use distinct acoustic models reflecting the distinct training data available for the two scenarios.

In addition to ASR, diarisation (who spoke when) is also included. This information is used to mark up the ASR output assigning utterances to speakers.

3. A Real-Time Architecture

3.1. Overview

Real-time ASR is by no means a new concept. It has been in use in commercial products for many years. The novelty here comes from the the fact that the system is distributed over computers and organisations and uses components from many vendors and authors.

The system architecture is built around a large vocabulary ASR decoder known as "juicer" [3]. However, both input and output are non-trivial. Input is obtained via a microphone array and beam-former over a TCP socket. Output is to a real-time database known as the "hub". Quite heavy use is made of (TCP) sockets, rather than more traditional file and screen input/output. The system is illustrated in figure 1 and described at some length in the remainder of this section.

3.2. Audio capture

The core audio device for the MDM system is a circular array with eight microphones. Coupled with the (typically four) IHM inputs, this presents a non-trivial audio acquisition task. We found that a convenient way to handle this is with a VST host[1]

[1]Plogue Bidule

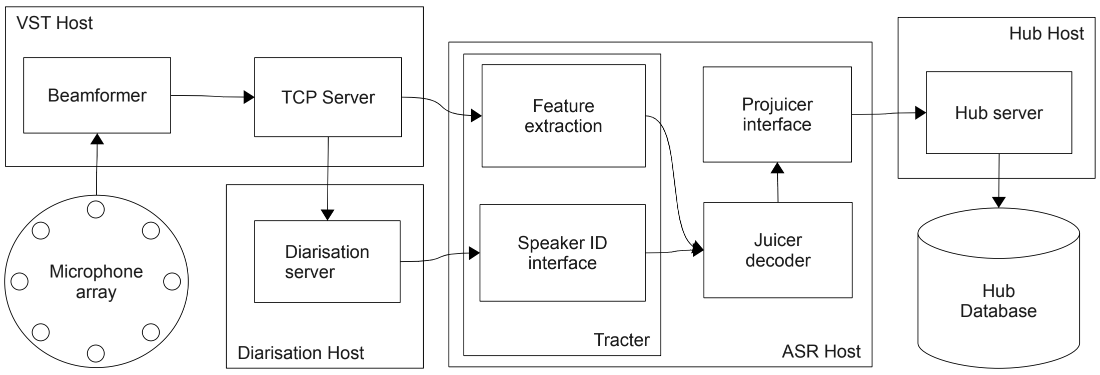

Figure 1: Block diagram of a typical meeting room real-time system distributed over four host machines: VST, diarisation, ASR and database. Hub consumers typically connect from other machines.

running on a Windows based PC. To this end, we wrapped the beam-former into a VST plugin. Further plugins act as servers, allowing processing modules requiring audio to connect over TCP and be supplied with audio at 48kHz.

The microphone array, described more fully in [4], has eight microphones in a circular arrangement of 20cm diameter. Initially, noise reduction is performed on each of the eight channels using a Wiener filter, with filter estimates generated from a prior recording from the room. A frequency domain delay-sum beam-former with post-filter, implemented as a VST plugin, is then directed at each of four locations known in advance to be those of the seated speakers. The signal corresponding to the direction with the highest energy is then forwarded to the socket connection.

The TCP solution also allows a very natural organisational interface. In this case, the audio acquisition was developed on Windows based PCs at one organisation, whereas the later stages were GNU/Linux based at other organisations.

3.3. Feature acquisition

The off-line AMI system comprised several passes, each using increasingly advanced features. These ranged from what might be called standard PLP (Perceptual Linear Prediction) features in the first pass, to tandem features with bottleneck MLPs (Multi-Layer Perceptrons) in later passes. Aside from the speech technology difficultly in reducing the number of passes, this also presented an engineering difficulty in combining the different feature extraction techniques.

The solution was a data-flow architecture known as "tracter". Data-flow is a well established signal processing technique that represents individual processing elements as vertices in a directed graph. The individual processing elements often run as separate processes or threads. Tracter is not (yet) a threaded system; data is propagated through the graph using a "pull" mechanism, instigated by the sink. The request from the sink is propagated back though the network, with each element in turn requesting enough data from its inputs to perform the operation. Pull mechanisms lend themselves to systems that do not necessarily run in real-time. In this case, it allows the data-flow to be driven by the recogniser, which in turn is the most CPU-intensive task. Whilst it runs *overall* in real-time, it is not mandated to do so, as it would be by a push mechanism. Tracter is hence also suitable for off-line use.

Tracter modules have been created to do audio acquisition from sockets, files and audio APIs, and to do basic feature extraction. In the context of the AMIDA system, tracter also contains components wrapping the following technologies:

- HTK, the HMM Toolkit, and in particular the HParm interface, enables standard features such as PLPs.
- BSAPI, the Brno Speech API, contains implementations of the tandem and fast VTLN techniques developed at BUT.
- Torch3, a machine learning library developed at Idiap[2].

3.4. Voice activity detection

VAD is implemented in tracter as a gate. Downstream from the gate, the decoder is unaware that VAD is happening; it just receives segmented data as if it were reading from a sequence of pre-segmented utterances (files). Upstream from the gate, however, the data is actually one continuous stream.

The gate segments the input stream based upon boolean speech / non-speech information from a VAD algorithm. The favoured algorithm is an MLP trained such that typical ASR features on the input layer appear as speech and silence outputs at the output layer. The VAD MLP is implemented using the Torch3 machine learning library. It is distinct from the MLP in the tandem features, which is implemented in BSAPI.

3.5. Diarisation

The system currently uses a Gaussian mixture based diarisation module from ICSI [5]. The module runs as a distinct process reading audio from the VST server and performing frame-level speaker identification. A tracter "SpeakerID" source converts time-stamped speaker ID requests from juicer into TCP requests to the diarisation module. The module responds immediately with a speaker ID. So, although the diarisation module runs as a separate module rather than a tracter component, it fits well into the pull-driven mechanism.

3.6. ASR decoder - Juicer

Juicer[3], is the core of the AMIDA ASR system. It is an HTK compatible Weighted Finite State Transducer (WFST) based token passing decoder. Juicer was originally used in the first passes of the AMI system as an off-line decoder, but was not *essential* as other tools existed to do a similar job. In particular, the early passes did not use large language models as these

[2]`http://www.torch.ch`
[3]Trans-*juicer*

could be used later in lattice rescoring. In the AMIDA system, however, a development push across three participating institutions has focussed on four areas:

1. Juicer is now architecturally a tracter sink. This means that any tracter graph can be used seamlessly for feature acquisition.
2. A new core decoder represents a speed up approaching an order of magnitude (depending upon the task). It is also capable of continuous decoding.
3. The WFST composition process has been overhauled to allow composition of very large transducers (see section 4).
4. A JNI (Java Native Interface) layer has been written to allow embedding in a Java process (see section 3.7).

The result is a faster and more capable decoder that can operate directly on high order language models in real-time. We use 4-gram, although higher order is possible. These capabilities mean juicer has become an essential part of the AMIDA system.

3.7. Output

The core of the AMIDA system, encompassing much more than just ASR, is a database known as the hub. The hub records XML encoded triples representing time-stamped events, and serves the information in real-time to applications. Processes supplying data to the hub are known as producers, and processes using information from the hub are consumers.

A simple example scenario might involve a face-recognition module producing meta-data associated with a face in an image (a name, say), and a browser consuming this information in order to annotate the image (with the name). In the context of the hub, ASR is a producer; it produces time-stamped words with speaker information.

The interface between ASR and the hub is also another organisational interface. In particular, a middleware suite allowing producers to send triples was written by the hub maintainers in java. Juicer is written in C++. To enable juicer to send hub triples and act as a producer, a JNI layer known as Projuicer has been written. Projuicer is able to collate ASR information along with other speech-related meeting metadata such as time-stamps, meeting ID and participant ID.

The system is designed to take time from a single source, in this case the VST host. Time stamps are propagated though the whole processing chain, enabling projuicer to time label all words. In practice, the utterance start and end times are reliable as they come from the VAD, but the word times are not reliable as they are subject to label pushing in the WFST.

4. Quantitative illustration

The real-time system uses the language models developed for the NIST RT evaluations [2]. These are typically 50,000 word N-gram models. Generally speaking, although the final recogniser will run on a 32 bit system, i.e., in under 4GB of memory, the WFSTs must be composed on a 64-bit system. The largest such system available to us at the time of writing has 32GB of core memory, allowing WFSTs up to around 30 million arcs. We use the FSM toolkit available from AT&T [6].

One of the most significant bottlenecks in the language model composition was associated with silence models [7]. In practice, we find that a good performance / size trade-off can be achieved by omitting the context free short pause models that can be freely inserted between triphones, but retaining context independent (monophone) silence models.

One of the main differences between the AMI and real-time systems is the language modelling. The AMI system used a two pass approach consisting of recognition with a 2-gram model followed by lattice rescoring with a 4-gram model. In AMIDA, juicer enables us to replace the two passes with a single pass using a single (slightly smaller) 4-gram model.

Figure 2 shows a comparison in terms of speed and error rate. The curves shown in the figure refer to a single pass juicer

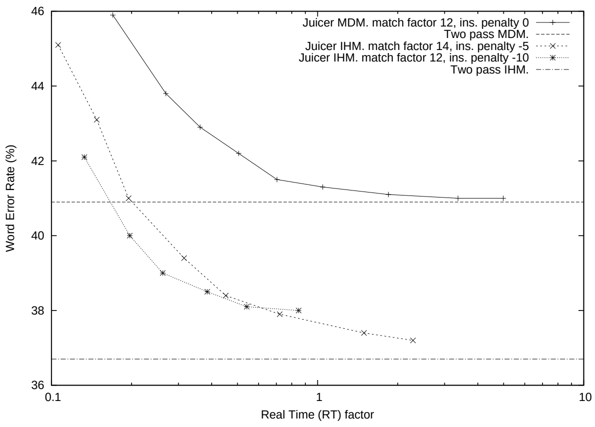

Figure 2: Speed vs. error rate for example IHM and MDM systems.

system evaluated on RT07 evaluation data. They are based on automatic joint optimisation of decoder parameters by simultaneous change of pruning parameters. For IHM, the two curves represent different combinations of insertion penalty and match factor. The horizontal lines represent the best (baseline) error rate obtained using the two-pass system based on the HTK decoder HDecode. This latter system runs in many times real-time. The plot shows that we can obtain close to asymptotic performance in real-time.

5. More advanced feature extraction and normalisation

During development, we have used an IHM acoustic model with a simple energy based VAD. The original model was a typical 13 dimensional PLP with energy and two orders of dynamic features to give 39 dimensions, trained on a corpus of meetings. This model was re-trained using a single-pass re-estimation on a front-end written using native tracter modules.

For the MDM system, however, we will use bottleneck tandem features [8] and an MLP based VAD. Such a system requires a significantly more involved feature extraction graph. This is illustrated in figure 3. The MLP VAD is implemented using Torch3, and the PLP and transforms are implemented using BSAPI.

The fast VTLN is an implementation of the technique described by Welling et al. [9]. MAP adaptation is used to create a Gaussian mixture for each warping factor, and the warping factor is smoothed by a single pole (forgetting factor) filter. Architecturally, the fast-VTLN module (and, to an extent, the filterbank module) incorporates a complete PLP extraction stage.

This illustrates a choice of granularity that exists when wrapping BSAPI, which itself deals with feature extraction

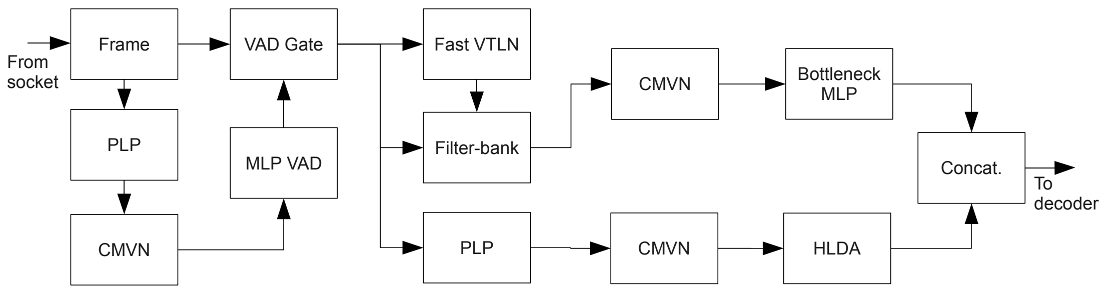

Figure 3: Simplified tracter graph of the MDM feature extraction incorporating BSAPI and Torch3 wrappers and native tracter components.

chains, and other libraries. When a function exists in a library, it is generally better to use it. For the more experimental modules, however, tracter allows components to be plugged and un-plugged easily. One such experimental component is the Cepstral Mean Variance Normalisation (CMVN). CMVN is normally implemented by using statistics accumulated over a known speaker. In a real-time system, however, this can be improved upon by accumulating in a recursive averaging manner. This on-line CMVN is implemented natively in tracter as a recursive form of that described in [10].

6. Conclusions

We have progressed the (off-line) AMI system towards an online system suitable for the real-time and distant access requirements of AMIDA. In doing so we have created a faster, more capable and demonstrable ASR system that is useful for downstream applications.

AMIDA is due to end in late 2009, by which time we will have a fully operational MDM system including downstream applications such as content linking and user engagement. AMIDA is, in principle, pursuing an open-source policy. At the time of writing, juicer and tracter are open-source and are freely available for download[4][5]. They are released under a BSD style licence.

7. Acknowledgements

Numerous people have contributed, with and without their knowledge, to the system described above. Juicer and its associated tools were originally written by Darren Moore and the beam-former by Iain McCowan. We are grateful to Pavel Matejka for the description of fast VTLN, to Petr Schwarz for BSAPI and to ICSI for the diarisation software.

This work was supported by the European Union 6th Framework Programme IST Integrated Project "Augmented Multi-party Interaction with Distance Access" (AMIDA, FP6-033812) and the Swiss National Center of Competence in Research (NCCR) on "Interactive Multi-modal Information Management" (IM2).

8. References

[1] A. Popescu-Belis, E. Boertjes, J. Kilgour, P. Poller, S. Castronovo, T. Wilson, A. Jaimes, and J. Carletta, "The AMIDA automatic content linking device: Just-in-time document retrieval in meetings," in *Machine Learning for Multimodal Interaction*, ser. Lecture Notes in Computer Science, A. Popescu-Belis and R. Stiefelhagen, Eds. Springer-Verlag, 2008, vol. 5237, pp. 272–283, 5th International Workshop, MLMI 2008.

[2] T. Hain, L. Burget, J. Dines, G. Garau, M. Karafiat, D. van Leeuwen, M. Lincoln, and V. Wan, "The 2007 AMI(DA) system for meeting transcription," in *Multimodal Technologies for Perception of Humans*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2008, vol. 4625, pp. 414–428, International Evaluation Workshops CLEAR 2007 and RT 2007, Baltimore, MD, USA, May 8-11, 2007, Revised Selected Papers.

[3] D. Moore, J. Dines, M. Magimai Doss, J. Vepa, O. Cheng, and T. Hain, "Juicer: A weighted finite-state transducer speech decoder," in *Proceedings of the 3rd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, 2006.

[4] M. Lincoln, I. McCowan, J. Vepa, and H. K. Maganti, "The multichannel wall street journal audio visual corpus (MC-WSJ-AV): Specification and initial experiments," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, November 2005, pp. 357–362, San Juan, US.

[5] C. Wooters and M. Huijbregts, "The ICSI RT07s speaker diarization system," in *Multimodal Technologies for Perception of Humans*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2008, vol. 4625, pp. 509–519, International Evaluation Workshops CLEAR 2007 and RT 2007, Baltimore, MD, USA, May 8-11, 2007, Revised Selected Papers.

[6] M. Mohri, F. C. N. Pereira, and M. Riley, "The design principles of a weighted finite-state transducer library," *Theoretical Computer Science*, no. 231, pp. 17–32, January 2000.

[7] P. N. Garner, "Silence models in weighted finite-state transducers," in *Proceedings of Interspeech*, September 2008, Brisbane, Australia.

[8] F. Grézl, M. Karafiát, K. Stanislav, and J. Černocký, "Probabilistic and bottle-neck features for LVCSR of meetings," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2007, pp. 757–760, Hononulu, US.

[9] L. Welling, S. Kanthak, and H. Ney, "Improved methods for vocal tract normalization," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, March 1999, pp. 764–764, Phoenix, US.

[10] O. Viikki and K. Laurila, "Noise robust HMM-based speech recognition using segmental cepstral feature vector normalization," in *Robust Speech Recognition for Unknown Communication Channels*. ISCA, April 1997, pp. 107–110, Pont-à-Mousson, France.

[4]`http://juicer.amiproject.org/juicer`
[5]`http://juicer.amiproject.org/tracter`