A Meeting Browser that Learns

Patrick Ehlen and **Matthew Purver** and **John Niekrasz**
Center for the Study of Language and Information, Stanford University
Cordura Hall, 210 Panama Street, Stanford, CA 94305
{ehlen, mpurver, niekrasz}@csli.stanford.edu

Abstract

We present a system for extracting useful information from multi-party meetings and presenting the results to users via a browser. Users can view automatically extracted discussion topics and action items, initially seeing high-level descriptions, but with the ability to click through to meeting audio and video. Users can also add value by defining and searching for new topics and editing, correcting, deleting, or confirming action items. These feedback actions are used as implicit supervision by the understanding agents, retraining classifier models for improved or user-tailored performance.

Introduction

Consider this snippet taken from the first few minutes of a meeting where the meeting leader, John, is attempting (with some difficulty) to type a shared agenda while simultaneously communicating his intended outline for the meeting to the other participants:

> John: The, uh, goal is to... um....
> Jim: Um, John...?
> John: Yes?
> Jim: Were you intending to put this in as a... as, uh, an agenda?
> John: Uh, yeah-yeah-yeah-yeah. I'm gonna let someone else do the.... I can't talk and type clearly!

In response to John's frustration at trying to type notes and talk at the same time, Jim takes over the task of typing the agenda:

> Jim: OK. Is this everything that's going to be in the agenda?
> John: Yes.
> Jim: OK.
> John: So while you're doing that, just let me give you a quick overview.

John's difficulty reflects a common conflict that befalls people in meetings – or any circumstance that calls for note-taking – since his activity demands verbal production to occur simultaneous to written production, resulting in a complex cognitive task. When John admits that his level of cognitive load is slowing him down, Jim offers to take over the task of creating the agenda and taking notes, arriving at a very natural collaborative solution. They now share the load, and their meeting continues at a much smoother pace.

But as natural as their new shared workload may be, even that solution isn't ideal, since Jim's task is now to take notes while listening to John, listening to the other participants, and participating in the meeting himself. This combination of duties results in another complex task that makes demands on Jim's cognitive subsystems of comprehension, selection, consolidation, and written production nearly all at once (Piolat, Olive, & Kellogg 2005).

Minimizing the effort required of these cognitive subsystems motivates our work on an intelligent Meeting Assistant, as part of the DARPA CALO ("Cognitive Assistant that Learns and Organizes") project. While our ambition is to eventually offload even the task of comprehension during meetings (perhaps to the chagrin of managers everywhere), at this point we can claim minor inroads in selection and, to some extent, consolidation, when it comes to detecting the action items assigned and locating the topics discussed during a meeting. These bits of information – the topics discussed and the tasks assigned – rank highly as things people would want to be able to access in a record of a meeting, whether they attended the meeting or not (Banerjee, Rosé, & Rudnicky 2005; Lisowska, Popescu-Belis, & Armstrong 2004).

Thus, the CALO Meeting Assistant analyzes multi-party speech and handwriting input from meetings in order to identify the topics and action items that might concern people after the meeting.

But even if we have a good general idea about what content to look for during meetings, recognizing and interpreting that content poses many problems. It's a difficult enough task for a human overhearer or eavesdropper who wasn't participating in the dialogue of a meeting to glean the same understanding of that dialogue as the participants themselves, thanks to impoverished grounding and audience design (Clark & Schaefer 1992; Schober & Clark 1989). Even professional minute-takers who are physically present at a meeting have difficulty selecting those items that the meeting participants themselves find significant (Whittaker, Laban, & Tucker 2005). That difficulty increases markedly when attempting to achieve understanding with a machine overhearer that uses noisy multi-party transcripts obtained from a speech recognizer.

One might argue that in fact there is no canonical interpretation of the contents of a meeting, and that different people can come away with widely different interpretations of what happened due to their different interests or requirements. They may be interested only in action items which concern them, or which relate to a particular project. Attempts to segment a meeting by "topic," in particular, seem to be subjective: People may be interested in segmenting by the activity performed or by the state of the meeting (Dielmann & Renals 2004; Reiter & Rigoll 2004; Banerjee & Rudnicky 2004), rather than specifically by the subject matter discussed (Galley *et al.* 2003; Gruenstein, Niekrasz, & Purver 2005). And interpretations of the subject matter itself can differ widely, leading to poor inter-annotator agreement on topic boundary placement, especially as the notion of "topic" becomes more fine-grained (Gruenstein, Niekrasz, & Purver 2005).

So a process of active learning is central to the CALO Meeting Assistant, stimulated by a meeting browser designed specifically to solicit feedback from meeting participants about the things our assistant believed it detected. That feedback is then used to improve and personalize the assistant's detection algorithms.

Automatic Understanding

The CALO Meeting Assistant integrates many technologies to analyze recorded meetings and extract useful information from them. To understand the components specifically related to extracting action items and topics, we should first look at a brief overview of the system architecture.

System Architecture

The Meeting Assitant is comprised of the following principal components: a recording architecture, a set of components which provide natural language analysis capabilities, a knowledge base, and a meeting browser. In this paper we discuss a subset of the language analysis components, and the meeting browser.

A meeting is recorded, processed, and browsed in the following manner: The meeting participants each wear headset microphones, connected to a personal laptop on which a note-taking application and other tools are provided. The audio signal and any other actions performed on the laptop are recorded and archived to a server for each participant. After the meeting is finished, a sequence of processing steps are executed on the recorded audio to produce multiple layers of analysis, including speech transcripts and topic segments. When the automatic analysis of the meeting is complete, the results are saved to a knowledge base on the server. This information is then made accessible as XML on a web server, which is interpreted and displayed by the meeting browser, an AJAX application in a browser. Using the browser, participants can view the meeting contents and replay recorded audio.

In the remainder of this section we will describe action item analysis, topic analysis, and the browser interface components in detail.

Action Item Identification

One way we hope to free up the cognitive effort of participants in a meeting is by automatically detecting and recording action items that are discussed, providing a list that helps people recall the tasks they agreed to do, and which can also be revisited at subsequent meetings to track their progress.

But what exactly is an action item? In our view, action items are specific kinds of decisions that are common in meetings, and occur when group responsibility for a concrete task is transferred to some particular person who assumes ownership of that responsibility. That person does not need to be the person who actually performs the assigned task, but engages in a social interaction that commits to seeing that the task will be completed; that is, that person becomes the owner of the action item. Since that action item is coordinated by more than one person, its initiation is reinforced by uptake among the owner and other participants that the action should and will be done, often resulting in a round of agreement. And to distinguish that action from more vague future actions that are still in the planning stage, an action item is often expected to be carried out within a specific timeframe, and that timeframe is made explicit.

These four components of an action item – a task description, an owner, a round of agreement, and an explicit timeframe – form the crux of our dialogical and hierarchical approach to action item detection, which differs from prior work on task detection. Prior work has focused on the level of individual sentences or utterances, classifying each as task-related or otherwise, whether dealing with sentences from e-mails (Corston-Oliver *et al.* 2004) or individual utterances from spoken dialogue (Gruenstein, Niekrasz, & Purver 2005; Morgan *et al.* 2006). For e-mail, these flat approaches have shown some success; but in spoken interaction, where much of the crucial information (e.g., ownership acceptance and overall agreement) comes from the dialogue structure itself as much as from the lexical content, performance seems limited (even when working on manual transcriptions rather than errorful speech recognition output). By ignoring the structure of the dialogue, flat approaches also lose information that can help us gain insight into the structure and properties of the action item itself (for example, the due date and person responsible), which is vi-

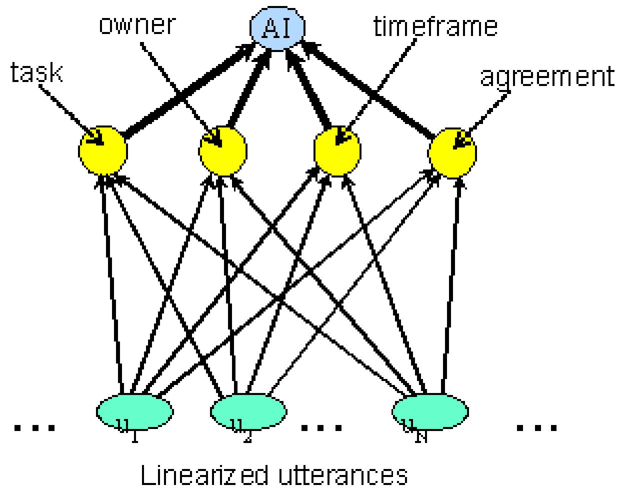

Figure 1: Defining an action item by classifying multiple utterances with dialog subclasses

tal when attempting to extract a useful representation of the action item for presentation to a user.

In contrast, our latest approach attempts to exploit a shallow notion of discourse structure, by looking for four separate subclasses of utterances which tend to be associated with action item discussion (see Figure 1). It looks across a window of multiple speakers and utterances, using a hierarchical combination of supervised classifiers to classify each utterance with any combination of these four subclasses. It then leverages the presence of multiple distinct subclass types to determine if the discussion in the window constitutes an action item. This approach improves accuracy beyond a flat approach, and also helps isolate information that facilitates a structured representation of the action item (see (Purver, Ehlen, & Niekrasz 2006)). Each sub-classifier is trained to detect a class of utterance which makes a particular discourse contribution to establishing an action item: proposal or description of the related task; discussion of the timeframe involved; assignment of the responsible party or owner; and agreement by the relevant people. An overall decision is then made based on local clusters of multiple discourse contributions, and the properties of the hypothesized action item are taken from the contributing utterances of various classes (the surface strings, semantic content or speaker/addressee identity, depending on the utterance subclass). Multiple alternative hypotheses about action items and their properties are provided and scored using the individual sub-classifier confidences.

Topic Identification

To answer the kinds of questions about meeting topics that users are likely to ask (Lisowska, Popescu-Belis, & Armstrong 2004), we require two elements: topic segmentation (dividing the discourse into topically coherent time segments) and topic identification (providing some model of the topics associated with those segments). These are joint problems, and we attempt to solve them as a joint inference problem. The meeting discourse is modeled as being generated by a set of underlying topics, with each topically coherent segment of discourse corresponding to a particular fixed weighted mixture of these topics. By using a variant of Latent Dirichlet Allocation (Blei, Ng, & Jordan 2003), we can then jointly learn a set of underlying topics together with a most likely segmentation (see Figure 2). Segmentation accuracy rivals that of other methods, while human judges rate the topics themselves highly on a coherence scale (see (Purver *et al.* 2006) for details).

Topics are learned over multiple meetings and are stored in a central topic pool. They can then be presented to the user as a summary (labeled via the top most distinctive words) to be used to guide audio and video browsing. They are also used to interpret a user keyword or sentence search query, by finding the weighted mixture of learned topics which best matches the words of the query and returns the most closely related segments or phrases.

User Interface

Research on multi-party dialogue in meetings has yielded many meeting browser tools geared toward querying and summarizing multimodal data collected from meetings (Koumpis & Renals 2005; Tucker & Whittaker 2004). Why create another? Existing tools focus on rapid information retrieval by users, and are designed to optimize the speed and success of finding answers to a broad class of queries. But the lack of linguistic metadata available to them results in a focus on the playback of signals rather than the display of dialogical or semantic features of the meeting.

Because our aim in the CALO Meeting Assistant project is to automatically extract useful information such as the topics and action items discussed during meetings, our meeting browser has a different goal. Not only do we need an end-user-focused interface for users to browse the audio, video, notes, transcripts, and artifacts of meetings, we also need a browser that presents automatically extracted information from our algorithms in a convenient and intuitive manner. And that browser should allow – even compel – users to modify or correct information when automated recognition falls short of the mark.

In fact, compelling users to provide feedback is essential to addressing the "overhearer understanding problem" that results from impoverished grounding. We deal with that understanding problem by designing agents to maintain multiple lexical and semantic hypotheses, and then rely on feedback from users about which hypotheses sound reasonable. But getting that feedback isn't always easy. A meeting browser that offers a high-level summary of the meeting's topics and action items is the ideal place to solicit feedback from end-users about what happened during a meeting. Our

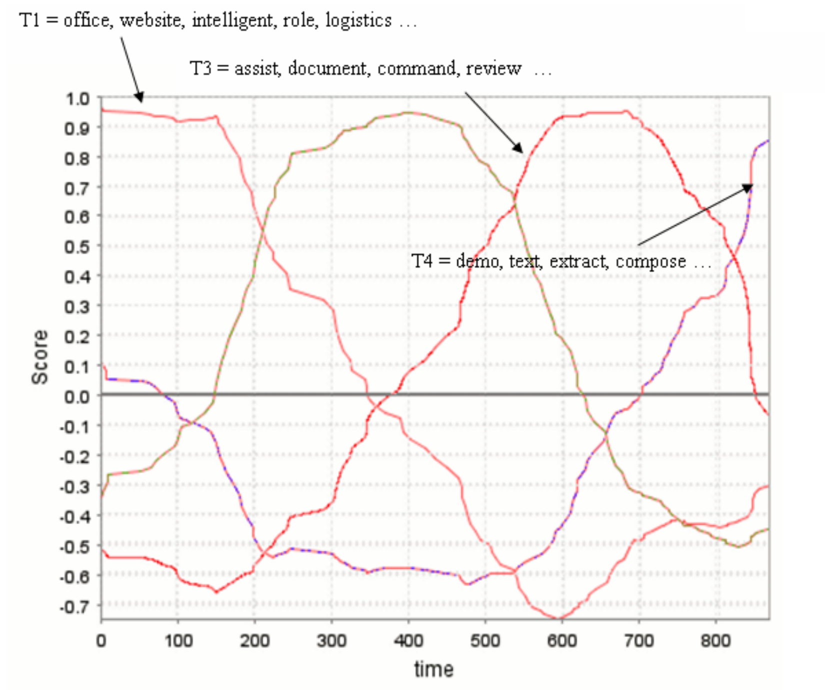

Figure 2: A meeting segmented into topic areas

browser interface exploits the transparency of uncertainty principle, and counts on a person's tendency to feel compelled to correct errors when those errors are (a) glaringly evident, and (b) correctable in a facile and obvious way.

Our approach is to present the meeting assistant's automatically extracted hypotheses alongside manual notes taken by meeting participants using the SmartNotes (Banerjee & Rudnicky 2006) meeting notes application. SmartNotes allows meeting participants to take shared notes during a meeting and then review them later in a simple web page. They also see automatically generated notes along with their own, and these hypotheses are highlighted at various degrees of illumination, according to the level of confidence given to the hypothesis. Hypotheses that have relatively high confidence scores have lighter highlighting, while hypotheses with lower confidence have darker highlighting. This way users can (a) easily spot the automatic hypotheses, and (b) get a quick sense of how likely those hypotheses are to be correct.

So for both manual and automated hypotheses, there is one coherent meeting review-oriented interface that allows the user to quickly make changes to text, delete unwanted items, and add items to a to-do list, creating a highly "personalized" representation of the meeting that is specific to that particular user's perception of the salient aspects of the meeting (see Figure 3). And these changes and feedback actions are saved in the database as a personalized "overlay" to the meeting, which will be loaded dynamically the next time the user wishes to browse the meeting, as well as contributing to a more personalized re-training of the models underlying the detection algorithms.

Action Item Display

A user can view action items detected from the meeting in the browser and drag them to a bin that adds the items to the user's to-do list. For the properties of action items – such as their descriptions, owners, and timeframes – the background colors of hypotheses are tied to their sub-classifier confidence scores, so less certain hypotheses are more conspicuous. These hypotheses respond to mouse-overs by popping up the most likely alternate hypotheses, and those hypotheses replace erroneous ones with a simple click. If an entire action item is rubbish, one click will delete it and provide negative feedback to our models. A user who just wants to make a reasonable action item disappear can click an "ignore this" box, which will still provide positive feedback to our model.

Automatically detected action items are also listed alongside the manual notes taken at the meeting, where the user can click a "transcript" button to obtain the machine-recognized transcript of those utterances that occur just before and after the detected action item. This provides the user with a clearer context of the action item, which could then be edited to better represent the actual task. Since the machine transcript can be errorful, the transcript button is supplemented with an "audio" button, which will play the actual audio displayed in the transcript.

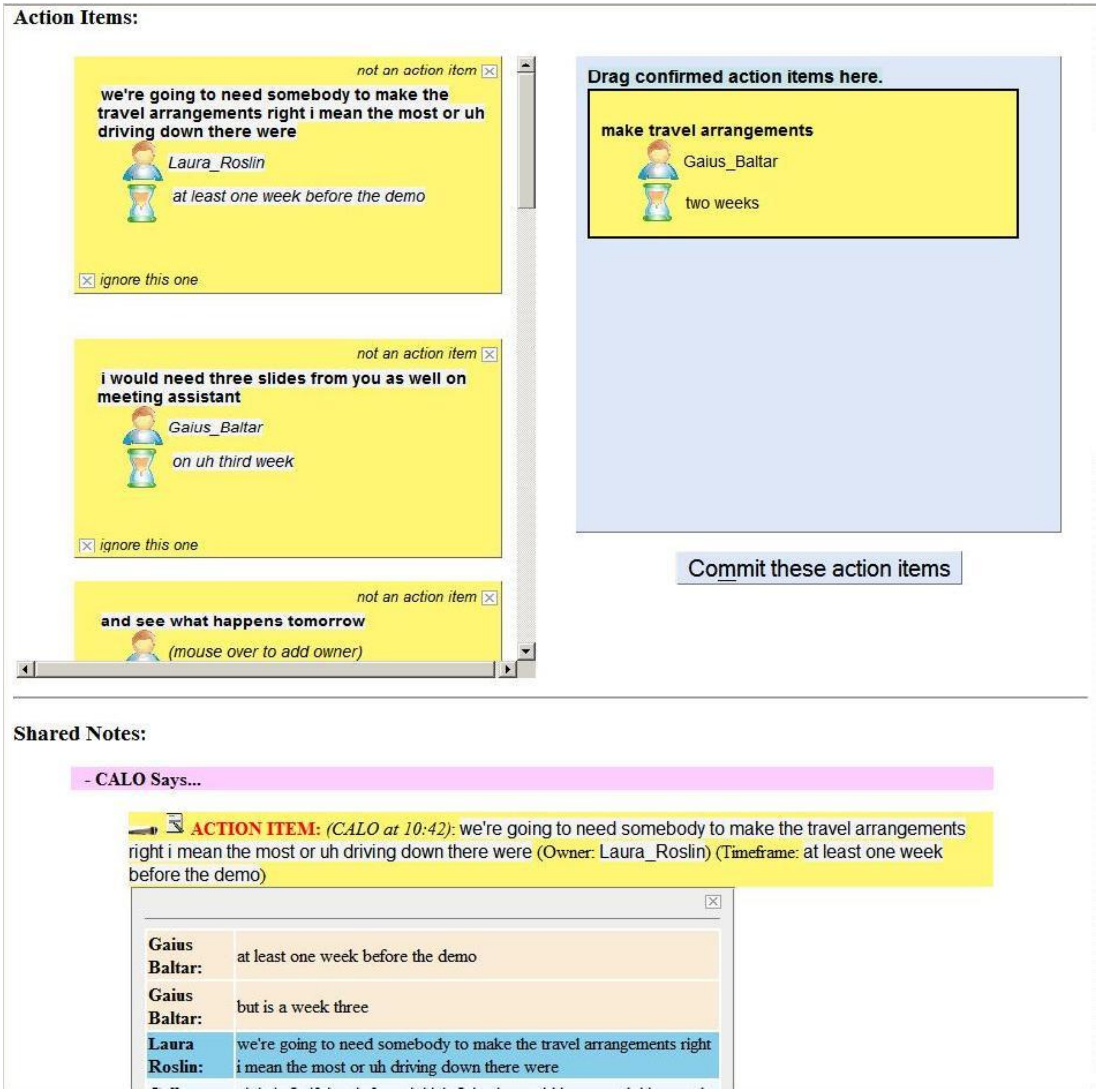

Figure 3: Meeting browser representation of a meeting, with manual and automatically generated action items

Topic Display

Topics appear as word vectors (ordered lists of words) for direct browsing or to help with user-defined topic queries. Given a user search term, the most likely associated topics are displayed, together with sliders that allow the user to rate the relevance of each list of words to the actually desired topic. As the user rates each topic and its words are re-weighted, a new list of the most relevant words appears, so the user can fine-tune the topic before the browser retrieves the relevant meeting segments.

This process, while requiring the extra "re-weighting" step from a straight-up search process, allows the user to define a highly customized set of topics, in addition to the topics already identified by our segmentation and identification algorithms. Those topics are saved in a topic pool which, after a personalized topic is defined, will subsequently apply to all searches across all meetings. While many topic detection routines assume that a pre-segmented and canonical list of topics will suffice for all users, our personalized topic pool assumes that users have their own subjective notions of what exactly a topic means (represented in our system by the vector of weighted words they create), and also that they will probably be concerned with similar topics for future meetings. These personalized topic pools could also be used to find similar topics in documents or e-mails.

Learning from Feedback

Recording Feedback

Monitoring and recording feedback is a non-trivial engineering problem. Our method records each user action as a set of changes to the underlying database of meeting events. These actions are stored independently of the original meeting record database, and the two are reinterpreted dynamically to produce a personalized representation of the meeting that is displayed in the browser. This gives learning and other post-processing algorithms universal access to the state of the browser at any given time in the use history. Feedback actions are timestamped and associated with individual users, so each user, by providing feedback, creates a personalized representation of the meeting, action items, and the topics discussed. Subsequent browsing sessions by a user then "re-applies" that user's feedback to the original meeting data, providing a personalized representation of what happened during a meeting, according to the data learned from feedback provided by the user.

Action Item Feedback

Feedback on automatically detected action items is obtained by giving users the opportunity to add each action item to their desktop to-do lists, (converted to a standard iCal representation of the task) which provides implicit feedback to our model that the detected set of utterances indeed signified a valid action item. Users can also "clean up" these action items before committing them to the to-do list, which provides additional feedback on the individual properties of the action item, and how the user believes these should be described.

The supervised action item classifiers can be retrained given utterance data annotated as positive or negative instances for each of the utterance subclasses (task description, timeframe, owner and agreement). We take user feedback as providing this annotation implicitly, inferring the individual utterance annotation classes by combining the feedback given with the stored links to the original utterances used as evidence to produce the hypothesis.

On the level of overall confirmation or deletion, this is reasonably straightforward. Confirmation of a hypothesized action item allows us to take the utterances used to provide its confirmed property values as positive instances for the subclasses associated with each property, while marking as negative instances all those utterances used to provide the alternate hypotheses not chosen. Conversely, deletion of a hypothesized action item allows us to mark all the utterances used to produce it as negative instances for all subclasses.

On the level of individual property feedback, things can become more challenging. Feedback which merely switches from one hypothesis to another is simple: we can mark the utterances corresponding to the accepted hypothesis as positive for the relevant subclass, and the others as negative. However, feedback which replaces all hypotheses with a new manually edited value is more of a challenge: We must find a relevant utterance (or set of utterances) in the nearby discourse, and make the assumption that it should be marked as the correct positive instance – however, this is a strong assumption and we must only make it in cases where our relevance measure scores highly. To assess relevance we can use the associated sub-classifier to score candidate utterances; use various similarity/distance metrics including simple lexical overlap and synonymy; treat this as an information retrieval problem and use standard measures like tf/idf; or a combination of all of these.

The most challenging feedback case is manual creation of a new action item (and use of manual typed notes taken during the meeting, which can be seen as a specific case of this). This requires us to search for likely relevant utterances for all properties, increasing the danger of misidentifying false positive instances. We can, of course, use the co-presence of apparently relevant utterances from multiple different subclasses to help increase our confidence. In the case of notes taken during the meeting, we can also use the timestamp to rule out any candidate utterances which occur after the note was taken.

In all cases, therefore, feedback provides implicit supervision, allowing us to automatically produce new training data so that we can re-train the classifier models for higher accuracy or user-specificity. Treating all users' feedback together allows us to produce a more robust general model; treating individual users' feedback separately could also allow us to produce tailored classifier models which reflect each user's preferences.

Topic Feedback

The topic extraction and segmentation methods are essentially unsupervised and therefore do not need to use feedback to the same degree. Yet even here we can get some benefit: As users define new topics during the search process (by moving sliders to define a new weighted topic mixture), these new topics can be added to the topic pool. They can then be presented to the user (as a likely topic of interest, given their past use) and used in future searches; they can potentially also be used to re-segment past meetings based on this new information, although this has not yet been investigated.

Future Directions

We've discussed our efforts here at creating a meeting assistant system that uses automatically generated transcripts from meeting recordings to detect action items and identify topics. This system is particularly unique in its use of user feedback – collected from meeting participants after the meeting using a meeting browser – to provide feedback to our detection models, making them more robust as well as more personalized.

How much of an impact does user feedback have on improving our models? This is a question we have just begun to investigate, and some preliminary findings on a small dataset look promising. Our next goal is to analyze feedback collected from the set of meetings that were collected using our system as part of the CALO Year 3 test process, and assess the relationship between user feedback and classifier performance. Even a simple analysis of how often people tend to provide feedback and what types of feedback they give could prove quite informative.

As we near the ability to achieve real-time performance for our detectors, we are also beginning to consider ways that feedback might be incorporated into a meeting assistant system that works during the meeting, and how much of a help or hindrance such a system might be. For instance, instead of soliciting feedback from users by asking them to browse a summary of the meeting after the fact, one possibility might be to provide a small applet that pops up an action item – on a laptop, phone, or PDA – shortly after the action item is discussed in the meeting, and allows a meeting participant to add it to a to-do list. But would this distraction be more cognitively taxing to a meeting participant than the simple act of writing the action item down on a notepad, as most people do today? And are these aspects of flow during a meeting that would be helped or hindered by the introduction of real-time meeting technology?

Finally, how will the behavior of people change during meetings when they know that the things they say are being recorded into a permanant transcript, and that the commitments they make are being automatically detected, recorded, and stored? Will people cease to take notes? Will they cease to make commitments? How will the process of decision-making change when this type of technology is introduced to the meeting room? Will the language they use become more specific? Or more vague? These are questions for which we are eager to discover answers.

References

Banerjee, S., and Rudnicky, A. 2004. Using simple speech-based features to detect the state of a meeting and the roles of the meeting participants. In *Proceedings of the 8th International Conference on Spoken Language Processing (INTERSPEECH - ICSLP)*.

Banerjee, S., and Rudnicky, A. 2006. Smartnotes: Implicit labeling of meeting data through user note-taking and browsing. In *Proceedings of the Human Language Techonolgy Conference of the NAACL, Companion Volume*.

Banerjee, S.; Rosé, C.; and Rudnicky, A. 2005. The necessity of a meeting recording and playback system, and the benefit of topic-level annotations to meeting browsing. In *Proceedings of the 10th International Conference on Human-Computer Interaction (CHI)*.

Blei, D.; Ng, A.; and Jordan, M. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Clark, H. H., and Schaefer, E. F. 1992. *Arenas of Language Use*. University of Chicago Press. chapter Dealing With Overhearers.

Corston-Oliver, S.; Ringger, E.; Gamon, M.; and Campbell, R. 2004. Task-focused summarization of email. In *Proceedings of the 2004 ACL Workshop Text Summarization Branches Out*.

Dielmann, A., and Renals, S. 2004. Dynamic Bayesian Networks for meeting structuring. In *Proceedings of the 2004 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Galley, M.; McKeown, K.; Fosler-Lussier, E.; and Jing, H. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, 562–569.

Gruenstein, A.; Niekrasz, J.; and Purver, M. 2005. Meeting structure annotation: data and tools. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*.

Koumpis, K., and Renals, S. 2005. Content-based access to spoken audio. *IEEE Signal Processing Magazine* 22(5):61–69.

Lisowska, A.; Popescu-Belis, A.; and Armstrong, S. 2004. User query analysis for the specification and evaluation of a dialogue processing and retrieval system. In *Proceedings of* *the 4th International Conference on Language Resources and Evaluation (LREC)*.

Morgan, W.; Chang, P.-C.; Gupta, S.; and Brenier, J. M. 2006. Automatically detecting action items in audio meeting recordings. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, 96–103. Sydney, Australia: Association for Computational Linguistics.

Piolat, A.; Olive, T.; and Kellogg, R. T. 2005. Cognitive effort during note-taking. *Applied Cognitive Psychology* 19:291–312.

Purver, M.; Körding, K.; Griffiths, T.; and Tenenbaum, J. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, 17–24. Sydney, Australia: Association for Computational Linguistics.

Purver, M.; Ehlen, P.; and Niekrasz, J. 2006. Detecting action items in multi-party meetings: Annotation and initial experiments. In *Proceedings of the 3rd Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*.

Reiter, S., and Rigoll, G. 2004. Segmentation and classification of meeting events using multiple classifier fusion and dynamic programming. In *Proceedings of the International Conference on Pattern Recognition*.

Schober, M. F., and Clark, H. H. 1989. Understanding by addressees and overhearers. *Cognitive Psychology* 21:211–232.

Tucker, S., and Whittaker, S. 2004. Accessing multimodal meeting data: Systems, problems and possibilities. In *Proceedings of the 1st Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*.

Whittaker, S.; Laban, R.; and Tucker, S. 2005. Analysing meeting records: An ethnographic study and technological implications. In *Proceedings of the 2nd Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*.