Browsing Meetings: Automatic Understanding, Presentation and Feedback for Multi-Party Conversations*

Patrick Ehlen, Stéphane Laidebeure, John Niekrasz,
Matthew Purver, John Dowding and **Stanley Peters**
Center for the Study of Language and Information
Stanford University
Stanford, CA 94305, USA
{ehlen, laidebeu, niekrasz, mpurver, jdowding, peters}
@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: new topics can be defined and searched for, and action items can be edited or corrected, deleted or confirmed. These feedback actions are used as implicit supervision by the understanding agents, retraining classifier models for improved or user-tailored performance.

1 Introduction

Research on multi-party dialogue in meetings has yielded many *meeting browser* tools geared toward providing visual summaries of multimodal data collected from meetings (Tucker and Whittaker, 2005). Why create another? Existing tools focus on facilitating manual annotation and analysis of abstracted knowledge, or on assisting the meeting process by allowing users to conveniently (but manually) add relevant information online.

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 artefacts 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.

2 Automatic Understanding

User studies (Banerjee et al., 2005) show that amongst the most requested pieces of information from a meeting are the *topics* discussed and *action items* established.

Action Item Identification. Our understanding suite therefore includes an agent for action item identification – see (Purver et al., 2006). We exploit a shallow notion of discourse structure, by using a hierarchical combination of supervised classifiers. 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 contributing utterances (the surface strings, semantic content or speaker/addressee identity). Multiple alternative hypotheses about action items and their properties are provided and scored using the individual sub-classifier confidences.

Topic Identification. Another agent splits meetings into topically coherent segments, providing models of the associated topics using vector space

This work was supported by DARPA grant NBCH-D-03-0010. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

brandial 2006: Proceedings of the 10th Workshop on the Semantics and Pragmatics of Dialogue, pages 173–174.

models. Topics are extracted as probability distributions over words, learnt over multiple meetings and stored in a central topic pool; they can then be used for audio/video browsing (labelled via the top most distinctive words) or to interpret a user keyword or sentence search query (by finding the weighted mixture of learnt topics which best match the words of the query).

3 User Interface

Agents that generate multiple hypotheses fare better with feedback from users about which hypotheses sound reasonable, but getting that feedback isn't always easy. A meeting browser is the ideal place to solicit feedback from end-users about what happened during a meeting. Our browser interface exploits the *transparency of uncertainty* principle, which counts on people's tendency to feel compelled to correct errors when those errors are (a) glaringly evident, and (b) correctable in a facile and obvious way.

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 mouseovers 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.

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.

4 Learning from Feedback

Action Item Feedback. The supervised action item classifiers can be retrained given utterance data annotated as positive or negative instances for each of the utterance classes (task description, timeframe, owner and agreement). User confirmation of a hypothesized action item allows us to take the utterances used to provide its properties as positive instances; conversely, deletion allows us to mark them as negative instances. Switching from one hypothesis to another for an individual property allows us to mark the utterances corresponding to the accepted hypothesis as positive, and the others as negative. Creation of a new action item, or manual editing or insertion of a property value requires us to search for likely utterances to treat as corresponding positive evidence; this can be done by using the relevant sub-classifier to score candidate utterances, and/or by string/synonym comparison, depending on the property concerned. Feedback therefore provides implicit supervision, allowing re-training models for higher accuracy or user-specificity.

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.

References

S. Banerjee, C. Rosé, and A. Rudnicky. 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*.

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

S. Tucker and S. Whittaker. 2005. Accessing multimodal meeting data: Systems, problems and possibilities. In S. Bengio and H. Bourlard (Eds.), *Machine Learning for Multimodal Interaction: First International Workshop, 2004*, v. 3361 of *Lecture Notes in Computer Science*, 1–11. Springer-Verlag.