Carnegie Mellon University
Research Showcase @ CMU

2007

Segmenting Meetings into Agenda Items by Extracting Implicit Supervision from Human Note–Taking

Satanjeev Banerjee
Carnegie Mellon University

Alexander I. Rudnicky
Carnegie Mellon University

Follow this and additional works at: http://repository.cmu.edu/compsci

Segmenting Meetings into Agenda Items by Extracting Implicit Supervision from Human Note–Taking

Satanjeev Banerjee
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15217, USA
banerjee@cs.cmu.edu

Alexander I. Rudnicky
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15217, USA
air@cs.cmu.edu

ABSTRACT
Splitting a meeting into segments such that each segment contains discussions on exactly one agenda item is useful for tasks such as retrieval and summarization of agenda item discussions. However, accurate topic segmentation of meetings is a difficult task. In this paper, we investigate the idea of acquiring implicit supervision from human meeting participants to solve the segmentation problem. Specifically we have implemented and tested a note taking interface that gives value to users by helping them organize and retrieve their notes easily, but that also extracts a segmentation of the meeting based on note taking behavior. We show that the segmentation so obtained achieves a P_k value of 0.212 which improves upon an unsupervised baseline by 45% relative, and compares favorably with a current state–of–the–art algorithm. Most importantly, we achieve this performance without any features or algorithms in the classic sense.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors, Algorithms.

Keywords: Implicit human supervision, meeting agenda segmentation, meeting note taking, SmartNotes.

INTRODUCTION
Meetings are the fora at which two or more people interact with each other, often through the speech medium, to discuss issues, present new ideas, make decisions, etc. Given how many meetings are held every day around the world, technologies that automatically assist humans during and after meetings can be very useful to meeting participants. Typical assistive tasks include:

- An automatic meeting note taker
- An automatic detector of decisions and action items
- An automatic summarizer of meetings

Performing each of these tasks automatically requires solving many difficult research problems, several of which are currently under investigation. These research problems include speech recognition in noisy meeting rooms [16, 11, 9], human activity recognition [15], automatic meeting summarization [12], meeting phase detection [2], action item detection [13], topic detection [7, 4, 14], etc. Each of these research problems are difficult to solve automatically, hence truly sophisticated meeting assistance technology has not made a large impact in meeting rooms around the world.

We are investigating an alternate approach to solving difficult research problems in the realm of automatic meeting assistance: Namely, by making intelligent use of the human meeting participants themselves. Humans are currently much better than automated systems at many tasks. For example, humans are much better than state-of-the-art automated systems at telling which topic is currently under discussion. At the same time, systems must be able to perform many of these tasks in order to provide higher level services. For example, a useful service, especially for people who missed a meeting, is to retrieve the segment of the meeting that contains discussions on a particular agenda item of interest to the listener [1]. Such a service would need to first segment a meeting into the agenda items.

In this paper we attempt to leverage the human meeting participants to help solve the problem of automatically splitting a meeting into segments, where each segment contains discussions on a single agenda item. Such a capability is useful for multiple higher level capabilities: As mentioned above, it is necessary for the playback or summarization of discussions on a specific agenda item. Meeting segmentation can also be useful for capabilities such as automatically reminding meeting participants that they are devoting more (or less) time than they had planned to an agenda item, automatically identifying the decisions and the action items, and who they should be assigned to, etc. To leverage humans, we employ a technique that we call *implicit human supervision*, which we discuss next.

IMPLICIT HUMAN SUPERVISION
Human supervision is routinely employed in building "intelligent" systems, most notably in the field of machine learning. Typically, humans are asked to perform the task that the system will eventually perform automatically, resulting

in *labeled data*. A system is then trained on this labeled data to model the human's actions. To ensure that such systems generalize to new "unseen" data, it is usually essential that the training data match the unseen data as closely as possible. This requirement in turn means that in general a large amount of data is necessary to cover the variations that can be expected in unseen data. However, collecting a large amount of data labeled by human beings is typically a very expensive task.

One hypothetical way of getting around this requirement for large amounts of data is acquiring the labeled data from the actual user of the system. A system trained on such user–provided data does not need to generalize to other users if the system is going to be primarily used by this particular user. Such an approach is adopted in training dictation systems: The user of the system provides a few hours of training data, and then the system is adapted to that particular user's speech patterns based on that data. (Such systems typically employ a hybrid approach, and also take advantage of large quantities of previously recorded speech data from other users).

However, this approach is generally infeasible because users are reluctant to provide training data for every task that a system wishes to perform. For example, it is inconceivable that meeting participants will be willing to manually segment their meetings into agenda items to provide the system with training data. Additionally, every time their agenda items or meeting participants change significantly, their previously provided labels may not remain useful any more.

This leaves us with the alternative that we call *implicit human supervision*: Acquiring supervision from the human user without requiring him to be aware of the fact that he is doing so. Such an approach does require two aspects to be true:

- There must be a *human task* - a task that the human wishes to accomplish.
- He must perform this task by interacting with the system.

If these requirements are met, then supervision can be acquired by designing the system's interface in such a way that as the human performs his own task, his actions result in labeled data as a byproduct, which can then be used to solve (or greatly help) the task that the system needs to perform.

Observe that no supervision can be obtained if the human does not use the system's interface to perform his task. Thus the system and the interface must provide the user with enough value to entice him to use this particular interface, thus constraining the design of the interface. This constraint in turn constrains the amount of supervision that can be extracted from the human's interactions with the system.

Our system's task is to split a meeting into segments, such that each segment contains discussions on a single agenda item. One potential human task in the context of meetings is that of taking notes at the meeting. Hence, we wish to provide meeting participants with a note–taking interface such that, as a participant takes notes in the interface, his actions result in labeled data that can be used to either directly produce a segmentation of the meeting, or to greatly help the
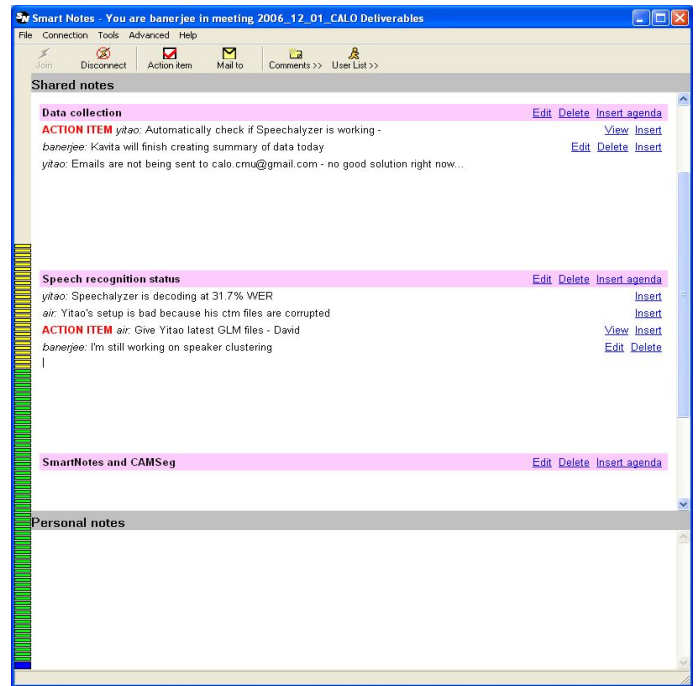

Figure 1: A screenshot of the SmartNotes client.

segmentation algorithm. Towards this end we have designed and implemented the SmartNotes system - a note taking and retrieval system that users can use to take notes at meetings, and access the notes afterwards. We describe this system and its interface in the next section.

THE SMARTNOTES SYSTEM
The SmartNotes system consists of two parts: A desktop application, called the "SmartNotes Client" through which participants can record their audio and type notes, and a web based application, called the "SmartNotes Website" through which participants can access the audio and notes, and write summaries of previous meetings. We describe these two components in the following subsections (and also in [3]).

The SmartNotes Client
The SmartNotes Client is a desktop application that meeting participants can use to take notes as well as record their speech. Each meeting participant is expected to come to the meeting with a laptop running the client on it, and with a close–talking microphone connected to the laptop. Figure 1 shows a screenshot of this application being used in a meeting.

Synchronization, Authentication and Joining a Meeting Before the meeting commences, each participant first starts the SmartNotes client. Each client synchronizes itself to a single pre-specified NTP machine. This ensures synchronicity between event timestamps created by different clients. Next, the user logs into the client by authenticating himself. The client authenticates the user by sending his username and password to a central "Meeting Server". The advantage of authenticating each user is that this allows us to trivially identify the speaker for each utterance, and the note taker for each line of notes written, instead of having to deduce these

facts through sophisticated methods. Once authenticated, the server sends the client the names of the currently running meetings; the user has the option of either joining one of these meetings, or "creating" a new meeting. Typically the meeting leader logs in first and creates a new meeting, and then the remaining meeting participants join the meeting. By joining the same meeting, users can share their notes, as described below.

Recording Speech   As soon as a participant joins a meeting, his audio starts getting recorded. The application has a "VU-meter" that continuously shows the user his volume level; meeting participants can use this VU-meter to adjust their microphone volume to an appropriate level. (This VU-meter is visible on the left margin of figure 1). Audio is recorded at 16khz, and uploaded to the Meeting Server opportunistically, based on network bandwidth availability. The transfer can continue beyond the end of the meeting, if sufficient network bandwidth is not available during the meeting. Additionally, the audio transfer is robust to network loss and power shutdowns.

Collaborative Note Taking   The main function of the Smart-Notes client is to allow meeting participants to take notes during the meeting. Once a user joins a meeting, he is shown the note taking interface, like in figure 1. This interface consists of two main note taking areas: The "shared notes" area and the "private notes" area. Notes typed into the shared notes area are automatically prepended with the author's name, and shared with all other meeting participants. This sharing is facilitated through the Meeting Server, and occurs in real time so that at all times, every meeting participant's shared notes look exactly the same. To avoid the problem of multiple simultaneous edits, participants can only edit/delete their own notes. Notes typed into the private area are not shared with any other meeting participant.

Agenda Based Note–Taking   The goals of this note taking interface are first to help meeting participants take notes, and second to automatically acquire labeled data to perform meeting agenda segmentation. To serve both goals, we allow participants to enter the agenda for the meeting into the interface. (Entering the agenda can be done at any time and by anyone, although typically it is done at the beginning of the meeting by the meeting leader). The interface splits the shared notes area (in each meeting participant's client) into as many text boxes as agenda items in the entered agenda, and labels each text box with the name of the corresponding agenda item. In figure 1, the agenda items are "Data collection", "Speech recognition status" and "Smart-Notes and CAMSeg". Meeting participants are expected to type notes on a particular agenda item in the box labeled with that agenda item's name. This interface design lets the user easily group his notes by agenda item so that he can later quickly retrieve all notes on a particular agenda item from multiple meetings. This design is also useful for acquiring data for topic segmentation: Every time a user enters a note under a particular agenda item, the system can conclude that at approximately that time the meeting participants were discussing that agenda item. This information can directly lead to a segmentation of the meeting, as we show below.

Action Item Identifier   In addition to taking notes, the meeting participants can also annotate certain notes as being "action items" - commitments by one or more participants to perform certain tasks within a specific time frame. When a user clicks on the "Action Item" button, he is provided with a form in which he can enter the details of the action item, such as what the action is, who is responsible for it, when it is due by, etc. The advantage to the group is that all the action items are separately available for future access, and can also be automatically emailed to participants as reminders before the next week's meeting.

The SmartNotes Website
We showed in [1] that meeting participants sometimes need to retrieve information from past meetings. To address these needs, the SmartNotes website allows meeting participants to access the speech and the notes taken in past meetings using the SmartNotes client. The main goal of the website is to help users quickly access information from previous meetings. It is hoped that by making information access easier, users may be enticed into using the SmartNotes client's interface in a way that gives the system better data. For example, once the users find the feature by which they can access notes from a single agenda item over multiple meetings useful, they may feel more encouraged to take notes within agenda item boxes in the SmartNotes client. Additionally the website provides more opportunities for acquiring data with which to improve other system goals besides topic segmentation; discussion of these goals is beyond the scope of this paper. However, for completeness, we shall briefly describe the SmartNotes website's interface.

Meeting participants log in to the website using the same username/password combination they use to log in to the SmartNotes client. Once logged in, the system shows the user a list of all his meetings. The user can click on any meeting to bring up all the shared notes taken during that meeting. Additionally, he can access the speech from each participant in that meeting, using the notes as an index. That is, the user can access n minutes of speech before and m minutes of speech after each note, from each speaker in the meeting (or he can listen to the combined audio channels). Finally, the user can create or access previously created summaries of the meetings.

BASELINE TOPIC SEGMENTER
Our goal is to segment the meeting time for a given meeting such that each segment contains discussions on a single topic. The notion of a *topic* is ill-defined; humans typically achieve low inter–annotator agreement when asked to segment texts and meetings into topics [8]. Since our ultimate goal is to retrieve sections of the meeting that belong to a particular agenda item, we use agenda items as the definition of *topic*. That is, given a meeting, our aim is to split the time from the start to the end of that meeting into segments, such that each segment contains discussions on a single agenda item. This formulation need not assume that the agenda is provided to the segmentation algorithm beforehand. However, it does assume that the meeting participants did follow an agenda during the meeting. We will show that this formulation does indeed result in high inter–annotator agreement.

Our baseline meeting segmentation algorithm is based on the TextTiling algorithm [10]. An in–depth description of our adaptation of this algorithm to the context of meetings is provided in [4]; here we give a brief overview of this algorithm. The algorithm's input is all the audio recorded during a meeting, along with the meeting's absolute start and end times. The algorithm's output is a set of time points within the meeting's start and end times that the algorithm considers to be times at which the meeting participants finished discussing an agenda item, and started discussing another one.

The algorithm proceeds by considering each time point t seconds from the start of the meeting, where t takes values 0, 1, 2, ... till the end of the meeting. For each such time point t, two windows are considered, one starting at time $t - k$ and ending at t and another starting at t and ending at time $t + k$. For each of these two k–seconds long windows, it constructs a vector containing the frequencies of the words uttered during the window by all the meeting participants, as output by a speech recognizer or as manually transcribed by a human, depending upon the experimental setup. Closed class words such as the articles and prepositions are ignored. Next the cosine distance between these two vectors is computed, according to the formula in equation 1.

$$cos(v_1, v_2) = \frac{\sum_{i=1}^{n} w_{i,v_1} w_{i,v_2}}{\sqrt{\sum_{i=1}^{n} w_{i,v_1}^2 \sum_{i=1}^{n} w_{i,v_2}^2}} \qquad (1)$$

Here, v_1 and v_2 represent the two vectors whose similarity is being computed, w_{i,v_1} represents the frequency of the ith word in vector v_1 (and similarly, w_{i,v_2} the frequency of the ith word in vector v_2), and n is the size of the vectors. Care is taken to ensure that each dimension in the two vectors represents the frequency of the *same* word in the two windows. Words that occur in one window but not in the other are considered to have a frequency of 0 in the other window. For time points near the beginning and the end of the meeting, we use smaller values of k to ensure that the windows to the left and right of any given time point have the same size.

The computed value quantifies how "similar" the word frequencies in the two windows are. Intuitively, the more dissimilar they are, the more likely it is that those two windows contain discussions on different agenda items, and consequently that the time point between those two windows is a topic boundary. After calculating the cosine similarity values, a *depth score* is computed for all the time points being considered in the meeting, as described in [10, 4]. Roughly speaking, depth scores are non–zero only for the bottoms of valleys and are higher if the valleys have "tall walls". Given these depth scores, a threshold is computed from them (the mean of the depth scores, plus half their standard deviation), and all time points with depth scores more than the threshold are reported as agenda item boundaries.

Observe that this algorithm is almost completely unsupervised; the only parameters that can be tuned are the size of the window (the value of k in the description above), and the threshold above which to report boundaries. We set the value of k to 350 seconds as this value performed well on separate held out data. As mentioned above, we follow [10] in computing the threshold, with the only difference being that we *add* half the standard deviation instead of subtracting it; adding performed better on our data.

The algorithm has no notion of the contents of different topics, and hence cannot be used to identify the topics for labeling purposes, for example. This fact is both its strength and weakness: While it does not need to be pre-trained on the specific topics in the meeting it needs to segment, its accuracy is low (as we shall see in the Evaluation section).

MAKING USE OF IMPLICIT SUPERVISION

As mentioned earlier, we are interested in improving upon the baseline segmentation results by making use of the implicit supervision provided by the meeting participants through their note taking in the SmartNotes note taking client. In this section, we describe three different ways we can take advantage of the implicit supervision.

Notes Based Segmentation

As described previously, meeting participants typically enter an agenda at the start of the meeting. The SmartNotes client then splits the shared note taking area into as many boxes as there are agenda items, each box labeled with the name of one of the agenda items. Meeting participants then type notes for each agenda item into the box labeled with the name of that agenda item. Further, as each note is entered, the system associates a time stamp with the note. Thus, for each note the interface captures several pertinent pieces of information:

- Which agenda item box it was typed into
- When it was typed
- Who typed it
- The text in the note

Although we can use all of these pieces of information to improve segmentation, in this paper we describe the simplest approach that only uses the first two pieces of information, that is, which agenda item box the note was typed into, and when it was typed. Table 1 shows these two pieces of information for each line of note in a hypothetical meeting. Using just these two pieces of information we can arrive at a segmentation of the meeting, as follows.

Note #	Time stamp (secs from start of meeting)	Agenda item box the note was typed into
1	100	Agenda item 1
2	150	Agenda item 1
3	170	Agenda item 2
4	230	Agenda item 2
5	250	Agenda item 1
6	290	Agenda item 3
7	350	Agenda item 3

Table 1: Time Stamps and Containing Agenda Boxes of Typed Notes in a Hypothetical Meeting

We first order the notes according to their time stamps. Next, for every pair of chronologically consecutive notes that were typed into different agenda item boxes, we hypothesize a boundary midway between the time stamps of those two notes. Thus, in the hypothetical example in table 1, the al-

gorithm does not hypothesize a boundary between notes 1 and 2, but does hypothesize one halfway between notes 2 and 3, that is at time point 160 seconds from the start of the meeting. Similarly, there would be no boundary hypothesized between notes 3 and 4, but there would be one halfway between notes 4 and 5 at time point 240 seconds, and again one halfway between notes 5 and 6, and so on. Thus, for this hypothetical example, the boundaries would be at time points 160, 240, and 270 seconds from the start of the meeting. That is there would be 4 segments in the meeting. We call this segmentation the *notes based segmentation*, since it is completely based on the notes taken by the meeting participants, and does not use the speech at all. Also note that we do not necessarily need to split the distance between the two notes in the different agenda item boxes precisely in half. For example, another reasonable candidate might be to split closer towards the earlier note. In fact, the location of the split could be empirically determined from data.

Note that this algorithm's accuracy completely depends on the note taking behavior of the participants in the meeting. For example, if the participants discuss an agenda item, but do not type even a single note on that item, then the notes based segmentation will not have a segment corresponding to that agenda item at all. Similarly, if one or more participants type notes into one agenda item box while the discussion is focusing on a different agenda item, then segmentation would also be incorrect. Finally, the quality of the notes-based segmentation will depend on the lengths of the gaps between consecutive notes that are in different agenda item boxes. The larger this gap is on average, the harder it is to tell where exactly the "real" boundary is. In the Discussion section we report on an experiment that quantifies the relationship between the number of notes taken and the performance of the notes based segmentation.

Modifying the Inputs to the Baseline Algorithm

Although we can derive a segmentation directly from the notes typed by the user, its accuracy may vary a lot depending on the note taking behavior of the specific group. However, instead of simply outputting the notes based segmentation, another strategy could be to use the notes to *improve* the baseline algorithm that uses the speech as its input, and is thus more *robust*.

The baseline algorithm assumes that all words are equally important for topic segmentation (except for closed class words as mentioned above). This is a reasonable assumption, if nothing further is known in advance about the meeting. However, if some (perhaps short) segments of the meeting are *labeled* with the agenda item they belong to, we can use these segments to learn which uttered words are indicative of the individual topics, and which are not.

We can derive such labeled data from the notes by performing almost the same process as that for deriving segmentation from the notes, with the following difference. For every pair of consecutive notes that are in two different agenda boxes, instead of hypothesizing one boundary halfway between the two notes, we hypothesize *two* boundaries, one each at the time points of the two notes. Thus, for the hypothetical example in table 1, the output labeled segments would be:

- Segment 1: Agenda item 1. From 100 to 150
- Segment 2: Agenda item 2. From 170 to 230
- Segment 3: Agenda item 1. From 250 to 250
- Segment 4: Agenda item 3. From 290 to 350

Thus, this algorithm takes the meeting time *between* consecutive notes that are in the same agenda box, and labels that time with the name of the agenda item of that box. However, when consecutive notes are in different boxes, it is unclear when the discussion changed from one agenda item to the next, so it leaves the entire meeting time between those two notes unlabeled.

Given these labeled segments, for every agenda item, we create a bag containing all the words uttered in all the segments labeled with the name of that agenda item. This gives us one bag of words per agenda item. From these bags, we find those words that occur in only one bag, and that occur at least twice. These are the words that most differentiate agenda items from each other. We then run the baseline algorithm exactly as before, but only using these words in every window pair, and ignoring all other words. We can use other measures of association; however, we started with the simplest algorithm possible.

Constraining the Outputs of the Baseline Algorithm

In the modified baseline algorithm described above, we first identify the most distinguishing words, and then run the same baseline algorithm as before. Although this improves the results compared to the original baseline (as shown in the Evaluation section), this algorithm still does not directly use any information from the notes, and thus often predicts boundaries that fall between notes that were typed into the same agenda item box. The simplest way to constrain these outputs is to simply only output boundaries that lie between consecutive notes that are in different agenda item boxes. We experiment with this simple constraint and report evaluation results below.

MEETING CORPUS

We have been using SmartNotes in our regular weekly project group meetings. These are all "natural" meetings, that is, these meetings would have taken place even if we were not using SmartNotes. (While the authors participate in these meetings, the remaining participants are not directly involved in the research reported in this paper). The form of the meetings is mostly report oriented, where each person in the group presents progress on his tasks since the last meeting. In addition to reports, questions are raised and discussed between team mates, and decisions are made about what next steps need to be followed the next week. Till the time of going to press (November 28th 2006), we have collected 26 meetings[1]. For the results in this paper we use only the first 10 meetings from this sequence. On average, each meeting is 31 minutes long, has 4.1 agenda items , and has 3.75 participants (ranging from 2 to 5). Each agenda item has on average 5.9 lines of notes in them, for a total of about 25 lines of notes per meeting. Note of course that these notes include *all* the notes taken by all the participants in the meeting in the shared note taking area.

---

[1]We plan to release this data to the research community; please contact the authors for more information.
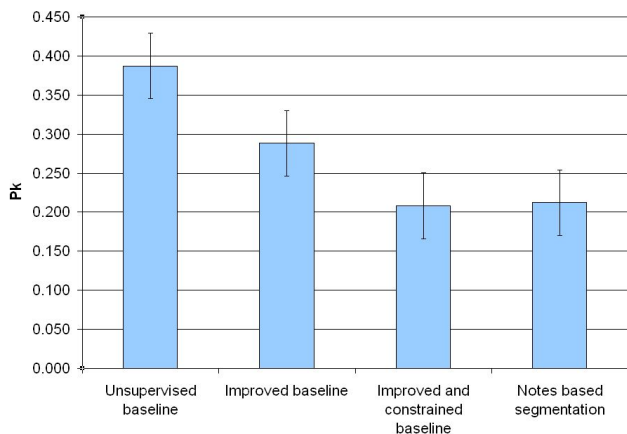
Figure 2: Overall performance for the four approaches, averaged over the 10 meetings in the corpus. Error bars are drawn using Standard Error.

Each meeting was manually segmented by two independent annotators. These annotators were provided with the agenda of the meeting (but not with the notes) and were asked to split the meeting into segments such that each segment corresponded to one of the agenda items in that list. We compute the agreement between the segmentation produced by the two annotators using the same metric (P_k) that we use to evaluate the segmentation algorithms. We describe this metric next, and the inter–annotator agreement in the following section.

EVALUATION
Evaluation Metric - P_k

We evaluate meeting segmentation using the P_k metric [6]. This metric computes the probability that two randomly drawn points a fixed time interval k seconds apart from the meeting are incorrectly segmented by the hypothesized segmentation, as compared to a reference segmentation, where a "correct" segmentation requires that both the reference and the hypothesis put the two points in a single segment, or both put them in different segments. Note that if the hypothesis and the reference put the two points in different segments, the hypothesis is deemed to be correct for those two points, even if the hypothesis and the reference predict different numbers of boundaries between the two points. Following [6], we compute the value of k for a given meeting to be half the average size of the segments in the reference segmentation for that meeting. Observe that P_k is a measure of *error*, and hence lower values are better. Specifically, P_k ranges between 0 (the reference and the hypothesis segmentations agree on every pair of points k seconds apart) and 1 (they disagree on every point).

Results

Figure 2 summarizes the overall results for the four segmentation approaches over the 10 meeting corpus. Note that for the algorithms that take spoken words as input ("Unsupervised Baseline", "Improved Baseline" and "Improved and Constrained Baseline" in the figure), the speech was automatically recognized using the CMU Sphinx–3 speech rec-

ognizer; across all the participants over the entire set of 10 meetings, the Word Error Rate was 45%. That is, 45% of the words automatically recognized were incorrect, as compared to human transcription. Although we do have manual transcriptions for our meeting corpus, we focus on segmentation results obtained using automatically transcribed speech since manual transcriptions will in general not be available. (However, see the Discussion section for segmentation results using manual transcriptions).

The baseline algorithm achieves an average P_k of 0.387 (standard deviation: 0.096), ranging from 0.272 to 0.543. This implies that in approximately 38.7% cases, the algorithm mis-segments a randomly drawn pair of points from the meeting. On the other end of the spectrum, the purely notes based segmentation achieves an average P_k of 0.212 (standard deviation: 0.099), ranging from 0.085 to 0.382. This result represents a 45% improvement over the baseline algorithm, and is a significant improvement ($p < 0.01$, using the Wilcoxon matched–pairs signed–ranks test).

Using the notes based segments to learn the utterance words that are most strongly correlated with the segments results in the "improved baseline" - the 2nd bar from the left in figure 2. This algorithm achieves a P_k of 0.288 (standard deviation: 0.148), ranging from 0.052 to 0.571. This result represents a 25% improvement over the unsupervised baseline, and the improvement is significant ($p < 0.01$ using the Wilcoxon test). The 3rd bar from the left in figure 2 represents the same improved baseline as above, but where the outputs of the algorithm are constrained so that there are no boundaries between chronologically consecutive notes that are in the same agenda box. This algorithm achieves an average P_k of 0.208 (standard deviation: 0.121), ranging from 0.060 to 0.423. This result represents a 46% improvement over the baseline, and is a significant improvement ($p < 0.01$, using the Wilcoxon test). Although this P_k value is slightly lower than that achieved by the notes based segmentation, the difference is not statistically significant.

DISCUSSION
Benchmarking Against [14]

To benchmark our meeting segmentation results against an independent state–of–the–art algorithm, we invited the authors of [14] to run their algorithm on our 10–meeting corpus. This algorithm uses an unsupervised generative topic modeling technique to segment multi–party discourses into topic segments. The algorithm was run on our corpus and, for each utterance in each meeting, it produced the probability that there was a change in agenda item at the end of that utterance. For each meeting, a threshold value was manually set, and all utterances with boundary probabilities greater than this threshold were marked as boundaries. A different threshold was set for each meeting to ensure that the algorithm produced exactly as many segments as the human annotator produced for that meeting; in a production system this threshold value can potentially be computed automatically based on the number of notes–based segments. This algorithm achieved an average P_k of 0.257 over the 10 meetings when run using the words output by the speech recognizer, and a P_k of 0.277 when using the manually transcribed

speech. Although both the purely notes based segmentation and the improved and constrained baseline outperform this segmentation (by 21% and 24% relative respectively), these differences are not statistically significant. Thus, we conclude that the notes based segmentation performs as well as a state–of–the–art segmentation algorithm.

The Quality of the Notes Based Segmentation
As noted above, the purely notes based segmentation performs exceedingly well – significantly better than the unsupervised baseline algorithm (45% relative improvement) and somewhat better than the state–of–the–art (although this difference is not statistically significant). This result shows the effectiveness of the idea of leveraging the human in the loop to implicitly acquire a solution to a problem. Observe that none of the meeting participants in these meetings were asked to split meetings into segments. They were simply given this interface and informed about the advantages (to *them*) of taking notes in the interface. The design of the interface produces a highly accurate segmentation directly from the users' note taking behavior. Further observe that the purely notes based segmentation was achieved without performing any speech recognition or feature extraction, or by applying any "sophisticated" algorithm in the classic sense.

One obvious drawback of this algorithm is that it is heavily dependent on the users' note taking behavior. For example, one would expect that if users took fewer notes than they did in our corpus, the notes based segmentation would suffer. To test the effect of a reduced number of notes on the performance of the notes based segmentation, we performed the following experiment. For each meeting, we randomly deleted X% of the notes (X varying from 0 to 100 with a step size of 10), performed the purely notes based segmentation using the remaining notes, and computed the P_k value. For each value of X, we repeated this experiment 1000 times, each time deleting a different random set of notes, and computed the average P_k over all meetings over all iterations.

Figure 3 shows a plot of the average P_k against the percentage of notes dropped. Observe that at $X = 0$, no notes are dropped, and the P_k is 0.212 as reported in the Evaluation section. At $X = 100$, all the notes are dropped, and no segments are created. The P_k value at this point is meaningless. At $X = 30\%$, the P_k value (0.249) is still better than that obtained by the state–of–the–art algorithm (0.257). Thus even with 30% less notes than that taken by the participants in our corpus, the notes based segmentation performs as well as the state–of–the–art algorithm. Further, even at $X = 70\%$, the P_k value (0.348) is better than that produced by the unsupervised algorithm (0.387). This implies that even if we have only 30% of the notes taken by the participants in our corpus, we can outperform the baseline unsupervised algorithm. These numbers are not surprising since the notes based segmentation algorithm only depends on how close the last note of one agenda item and the first note of the next agenda item are to the actual boundary between the two agenda items.

Other ways in which the quality of the notes based segmentation can be compromised include situations where users type notes on one agenda item while they are discussing another agenda item (either erroneously, or on purpose). We did not
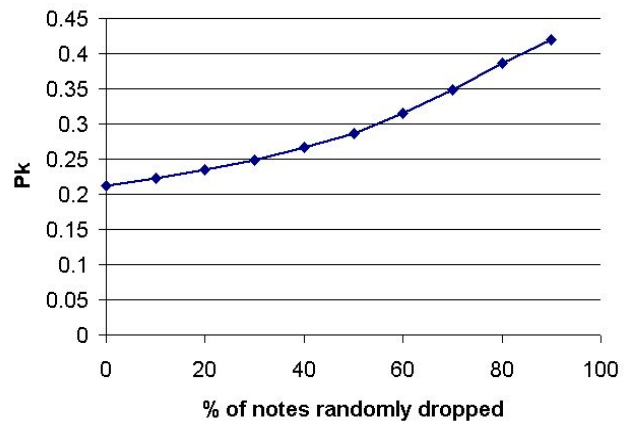

Figure 3: Performance of Notes Based Segmentation after Deleting a Random X% of the Notes.

experiment with this situation, but in general plan to test the notes based segmentation algorithm on different groups of people to assess its efficacy with varying human note–taking habits.

The Quality of the Other Algorithms
The "Improved baseline" that uses the notes based segmentation to learn the words that are correlated with the segments did not perform as well as we had hoped. This is perhaps because of a data sparcity problem, which can potentially be alleviated by accumulating the learning over multiple meetings which we shall attempt in the future. Using the notes based segmentation to constrain the "learned" segmentation performed as well as the notes based segmentation itself, but not better as we would have hoped. Perhaps a better approach would be to also look for "cue phrases" (topic independent phrases that signal topic boundaries, such as "moving on..."), as has been done by [6, 7].

Using Manual Speech Transcriptions
As mentioned earlier, our focus is on evaluating our various agenda segmentation algorithms using automatically recognized speech. However, to see the effect of the errors in such automatic transcripts on segmentation performance, we also ran all our experiments on manually transcribed speech. For the "Unsupervised Baseline" algorithm, the P_k improved from 0.387 to 0.339 by using the manual transcriptions, and this improvement was moderately significant ($p < 0.05$ using the Wilcoxon test). For the "Improved Baseline" algorithm, the difference was not statistically significant. For the "Improved and Constrained Baseline" algorithm, the P_k value improved from 0.208 to 0.152 by using manual transcripts, and this improvement was statistically significant ($p < 0.01$ using the Wilcoxon test). Finally, the algorithm described in [14] did not experience a statistically significant improvement in segmentation performance by using the manual transcripts. It is unclear why the "Improved and Constrained Baseline" algorithm benefited so much from the manual speech transcripts whereas the "Improved Baseline" algorithm did not. Further experiments are needed to understand the cause of this phenomenon.

Inter–Annotator Agreement

As mentioned above, each meeting in the corpus was manually segmented into agenda items by two independent human annotators. To compute the degree of agreement between their annotations, we follow [8] and simply compute the P_k between the two annotations. Since the choice of the reference affects the value of k (the distance between the two probe points in the calculation of P_k), we chose to perform this computation twice, each time using a different person's annotation as the reference. The resulting average P_k value over the 10 meetings was 0.062 (standard deviation: 0.049), regardless of the choice of reference annotation. These agreement numbers show that there is a lot of room for improvement for the automatic segmentation algorithms. Additionally, these agreement numbers are also substantially better than those reported in [8]. Several factors may have contributed to these high levels of agreement. First the concept of an agenda item is better defined than that of a "topic" as defined in [8]. Second, the meeting participants in our corpus typically displayed a somewhat disciplined adherence to the agenda at hand, unlike many of the meetings annotated in [8]. Finally, and perhaps most crucially, our annotators had access to the list of agenda items in the meeting they were annotating, whereas the annotators in [8] were asked to *identify* the topics in addition to segmenting the meetings into the topics, which can introduce further variability between annotators.

RELATED WORK

Implicit Human Supervision

Leveraging humans "in the loop" has been investigated in various formulations in the past, such as the work on Human Computation, most popularly exemplified in the ESP Game [17]. In this research, the notoriously difficult task of labeling images is transformed into a web–based two–player game in which randomly chosen pairs of humans try to associate the same image with the same label at the same time. People play the game because it is fun, and as a by–product a difficult research problem gets solved. Additional games to create boundary boxes for objects in an image [19] and to collect common–sense facts [18] have also been implemented. Our work is similar in that we create an interface such that as the humans perform their own task (note taking), they unwittingly solve the system task (meeting segmentation). The difference lies in the fact that in our case, the design of the interface is constrained by the human task, and we cannot "invent" new tasks the way new games are invented.

There is a similarity between implicit human supervision and the field of active learning where systems seek labels for informative data points from a human. Unlike active learning though, we assume that the human is *un*willing to perform labeling, so we must get the labels implicitly.

Meeting Topic Segmentation

As mentioned above, our baseline algorithm is a straight–forward adaptation of the classic Marti Hearst TextTiling algorithm [10] to the domain of meetings. Other approaches to topic segmentation include that of Beeferman et. al. [6] who use adaptive language models and "cue phrases" (phrases that typically occur near topic boundaries) to segment news transcripts into separate stories. Their application area is different from ours in that the topics discussed at a meeting are likely to be more strongly related to each other than stories in a newscast. Barzilay and Lee [5] present an HMM based method for learning models of topics and topic transitions in a specific domain from example texts in that domain. Closer to our application area is work done by Galley et. al. [7]. Their goal is to find topic segments in meetings by first finding chains of repeated words that overlap between adjacent windows of a meeting, and by training a decision tree classifier that uses features such as silences, speaker turns, cue phrases etc. Finally, as mentioned earlier, Purver et. al. [14] present an unsupervised generative model that learns topic models from unlabeled data.

CONCLUSIONS

In this paper we have explored the general idea of solving a difficult system task by acquiring implicit supervision from the actions of the human users of the system, as they interact with the system to perform their own tasks. We have applied this paradigm to the task of automatically splitting a meeting into segments such that each segment contains discussions on a particular agenda item. To perform this task, we have designed a note taking interface such that as the human meeting participants take notes in a meeting, their actions result directly in a segmentation of the meeting. We have evaluated this segmentation on a 10 meeting corpus, and have shown that we can achieve a P_k of 0.212, which represents a 45% relative improvement over an unsupervised baseline, and also compares favorably with segmentation produced by a state–of–the–art algorithm on the same data. Further, we have attempted to learn to improve the baseline segmentation from this notes–based segmentation, and have shown that such learning results in a modest improvement over the baseline algorithm.

FUTURE WORK

There are many interesting future directions for this work. First, as mentioned earlier, the notes based segmentation does not perform well if there are not enough notes in a meeting. One way to get around this problem is to train a classical topic segmenter using the notes based segmentation from meetings where there are a large number of notes, and applying the segmenter to segment meetings that have fewer or no notes. Second, we plan to evaluate our algorithms on different groups of people with different note taking habits to quantify the quality of the implicit supervision that can be derived from note taking. More generally, we would like to pursue more experimentation on the general idea of solving system tasks by acquiring supervision from a human task. For example, we can attempt to try and automatically detect those times in the meeting at which participants are likely to take notes, based on users' observable note taking behavior.

ACKNOWLEDGMENTS

REFERENCES

1. S. Banerjee, C. Rose, and A. I. Rudnicky. The necessity of a meeting recording and playback system, and the benefit of topic–level annotations to meeting browsing. In *Proceedings of the Tenth International Conference on Human-Computer Interaction*, Rome, Italy, September 2005.

2. S. Banerjee and A. I. Rudnicky. 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 2004 – ICSLP)*, Jeju Island, Korea, 2004.

3. S. Banerjee and A. I. Rudnicky. SmartNotes: Implicit labeling of meeting data through user note-taking and browsing. In *Proceedings of the Conference of the North American Association of Computational Linguistics - Human Languages Technology (NAACL-HLT) - Demonstration Track*, New York, NY, June 2006.

4. S. Banerjee and A. I. Rudnicky. A TextTiling based approach to topic boundary detection in meetings. In *Proceedings of the Ninth International Conference on Spoken Language Processing (Interspeech 2006 - ICSLP)*, Pittsburgh, PA, September 2006.

5. Regina Barzilay and Lillian Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Proceedings of the Main Conference*, pages 113–120, 2004.

6. D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177 – 210, 1999.

7. M. Galley, K. McKeown, E. Fosler-Lussier, and Hongyan Jing. Discourse segmentation of multi–party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 562 – 569, Sapporo, Japan, 2003.

8. A. Gruenstein, J. Niekrasz, and M. Purver. Meeting structure annotation: Data and tools. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, Lisbon, Portugal, September 2005.

9. T. Hain, J. Dines, G. Garau, M. Karafiat, D. Moore, V. Wan, R. Ordelman, and S. Renals. Transcription of conference room meetings: An investigation. In *Proceedings of Interspeech 2005*, Lisbon, Portugal, September 2005.

10. M. Hearst. TextTiling: Segmenting text into multi–paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, 1997.

11. F. Metze, Q. Jin, C. Fugen, K. Laskowski, Y. Pan, and T. Schultz. Issues in meeting transcription – the ISL meeting transcription system. In *Proceedings of the 8th International Conference on Spoken Language Processing (Interspeech 2004 – ICSLP)*, Jeju Island, Korea, 2004.

12. G. Murray, S. Renals, and J. Carletta. Extractive summarization of meeting recordings. In *Proceedings of Interspeech 2005*, Lisbon, Portugal, September 2005.

13. Matthew Purver, Patrick Ehlen, and John Niekrasz. Shallow discourse structure for action item detection. In *Proceedings of the HLT-NAACL workshop 'Analyzing Conversations in Text and Speech'*, New York, NY, June 2006.

14. Matthew Purver, Konrad Körding, Thomas Griffiths, and Joshua Tenenbaum. 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)*, pages 17–24, Sydney, Australia, July 2006. Association for Computational Linguistics.

15. Paul E. Rybski and Manuela M. Veloso. Using sparse visual data to model human activities in meetings. In *Workshop on Modeling Other Agents from Observations, International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2004.

16. A. Stolcke, C. Wooters, N. Mirghafori, T. Pirinen, I. Bulyko, D. Gelbart, M. Graciarena, S. Otterson, B. Peskin, and M. Ostendorf. Progress in meeting recognition: The ICSI–SRI–UW Spring 2004 evaluation system. In *NIST RT04 Meeting Recognition Workshop*, Montreal, 2004.

17. Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, New York, NY, USA, 2004. ACM Press.

18. Luis von Ahn, Mihir Kedia, and Manuel Blum. Verbosity: a game for collecting common-sense facts. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 75–78, New York, NY, USA, 2006. ACM Press.

19. Luis von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: a game for locating objects in images. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64, New York, NY, USA, 2006. ACM Press.