A TextTiling Based Approach to Topic Boundary Detection in Meetings

Satanjeev Banerjee and *Alexander I. Rudnicky*

Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA. United States
{banerjee,air}@cs.cmu.edu

Abstract

Our goal is to automatically detect boundaries between discussions of different topics in meetings. Towards this end we adapt the TextTiling algorithm [1] to the context of meetings. Our features include not only the overlapped words between adjacent windows, but also overlaps in the amount of speech contributed by each meeting participant. We evaluate our algorithm by comparing the automatically detected boundaries with the true ones, and computing precision, recall and f–measure. We report average precision of 0.85 and recall of 0.59 when segmenting unseen test meetings. Error analysis of our results shows that although the basic idea of our algorithm is sound, it breaks down when participants stray from typical behavior (such as when they monopolize the conversation for too long).

Index Terms: topic detection and segmentation, meeting understanding, spoken language understanding

1. Introduction

Our goal as a part of the CALO project[1] is to automatically understand discussions at meetings. A first step towards such understanding is to detect the topics of discussion. This problem can be broken into two parts – detecting *when* there is a change of topic, and determining *what* the topic is. In this paper we describe our current work on the first question – the detection of boundaries between different topics of discussion in meetings. While it is difficult to precisely define what a topic is, in this paper we consider all the discussions that pertain to a single agenda item to be part of one "topic". Thus our goal is to split a meeting into segments such that each segment belongs to an agenda item.

We do not expect to know the specific domain of the discussions in the meeting. Hence, we wish to avoid a training intensive algorithm. We base our algorithm on Marti Hearst's TextTiling [1] algorithm where the probability that a point in a text essay is a topic boundary is computed based on the similarity between the words in windows to the left and right of that point. This algorithm makes very little assumptions about the domain of the text being segmented, which makes it well suited to our application.

Other approaches to topic segmentation include that of Beeferman, et. al. [2] 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 [3] present an HMM based method for learning models of topics and topic transitions in a specific domain from example texts in that domain. Since we do not expect to have access to example meetings in the test domain, this algorithm is not suitable for our goal. Closest to our application area and approach is work done by Galley et. al. [4]. Their goal is to find topic segments in meetings by first finding chains of repeated words that overlap between adjacent windows of meeting, and by training a decision tree classifier that uses features such as silences, speaker turns, cue phrases etc. We use similar feature sets (word overlaps between adjacent windows; speech lengths from different participants) but avoid the training–intensive framework.

We describe our topic detection algorithm in the next section, followed by evaluation of the algorithm and analysis of the evaluation results. We conclude with some future steps.

[1] http://www.ai.sri.com/project/CALO

2. Topic Boundary Detection Algorithm

Our algorithm to detect topic boundaries in meetings is based on the TextTiling algorithm [1], which is an edge detection algorithm applied to topic segmentation in text. We adapt this algorithm to the context of multi–participant meeting conversation, as follows:

Creating Meeting Windows: We denote the time the meeting started as *t*, and consider candidate topic boundaries at 1 second intervals from *t*: at time *t*, *t + 1s*, *t + 2s*, etc. For each candidate boundary in a meeting, we compute the similarity between the speech in a pair of windows of time immediately preceding and following the boundary. For each run of the algorithm we use a fixed window length, specified in seconds. For example, if our window length is 15 seconds, and we are considering the candidate boundary at time *t + 35s*, we compute similarity between the window that starts at time *t + 20s* and ends at time *t + 35s* with the window that starts at time *t + 35s* and ends at time *t + 50s*.

Feature Extraction and Cosine Similarity: Similarity is computed by first extracting a vector of features from the speech in each window, and then computing the cosine similarity between the two vectors according to the following formula:

$$cos(v1, v2) = \frac{\sum_{t=1}^{n} w_{t,v1} w_{t,v2}}{\sqrt{\sum_{t=1}^{n} w_{t,v1}^2 \sum_{t=1}^{n} w_{t,v2}^2}} \quad (1)$$

where the two vectors are denoted by $v1$ and $v2$ respectively, $w_{t,v1}$ is the value of the t^{th} index of vector $v1$ (and $w_{t,v2}$ the t^{th} index of vector $v2$) and n is the size of each vector. Although this cosine similarity formula is not affected by scale (that is $cos(v1, v2) = cos(kv1, lv2)$ where k and l are constant scalar multipliers), it is affected by the range of values of each index in the vectors. For example, if $w_{0,vi}$ (i = 1 or 2) is in the range [0,1],

and $w_{1,vi}$ (i = 1 or 2) is in the range [100,200], the cosine similarity value will be much more sensitive to changes in index 1 of the vectors than to changes in index 0 of the vectors. One strategy to combat this problem is to normalize each dimension so that the values for each index are always within the same range (say 0 to 1). This strategy is problematic because it treats each dimension as being equally important. Our current approach is to apply the algorithm to the same meeting multiple times, each time with a group of "similar" features that have values in a similar (but not exactly the same) range. Thus, for each feature group, we obtain a separate "boundary predictor", and then linearly combine the boundary predictions from each predictor. This approach has the added advantage that we can weigh each boundary predictor differently according to how much importance we wish to attach to a specific feature group. Further, this allows us to integrate boundary predictions output by TextTiling type algorithms with those output by other boundary prediction algorithms. We smooth the output similarity values using an *unweighted sliding-average smooth*, with *smooth width* = 3. Note that the cosine similarity values are in the range [0,1], with higher numbers indicating higher similarity between adjacent windows, and thus lower likelihood that the time point in question is a topic boundary.

From Similarity Values to Boundary Predictions: To compute the probability that a time point is a topic boundary, we consider only those time points whose similarity valleys represent a "valley", that is, those time points whose similarity values are less than those at the immediately preceding and following time points. Following [1], we compute a *depth score* for each such valley point as follows. We first find the nearest "peak" time points preceding and following the valley point. Like a valley, a peak is a time point whose similarity value is greater than the similarity values of time points immediately preceding and following it. Denote the valley time point as v and the nearest peak points preceding and following v as p_1 and p_2. The depth score at time point v is then calculated (like in [1]) according to the formula:
$depthscore(v) = (sim(p_1) - sim(v)) + (sim(p_2) - sim(v))$
where $sim(t)$ represents the similarity value computed for time point t. Observe that the depth score depends on the relative depth of the valley compared to its closest peaks, and not on the absolute similarity value at the valley. Thus this formula considers a sharp drop in similarity value as more indicative of a boundary than a gentler drop, even though the actual similarity value may be high. Thus this formula can detect topic shifts even in a meeting where the overlap of words between different topics is high. Every non–valley time point in the meeting is given a depth score of 0.

The last step for the algorithm is to report a set of boundaries from the depth scores computed above. To do so we first compute a cut–off value (as in [1]) by subtracting the standard deviation from the mean of all the depth scores. We then consider only those time points whose depth scores are greater than this cut–off. We consider each such time point in descending order of depth score, greedily classifying it as a topic boundary as long as it does not occur within a fixed interval of a previously classified topic boundary. (This interval, which we call the "minimum topic length" is a trainable parameter). Once all time points whose boundary prediction values are above the cut–off have been considered, we output the set of chosen boundary time points.

2.1. Speech Activity Based Boundary Prediction

The speech activity based boundary predictor uses a TextTiling based approach to compute the similarity between equal sized windows immediately preceding and following each candidate topic boundary time point (as described above). The features used to compute the cosine similarity between the two windows are the lengths of speech for each participant in each window. Thus for each window a vector is created that has as many dimensions as there are participants in the meeting. Each dimension in the vector corresponds to the length of speech produced by a specific participant in the window under consideration.

Our motivation for using this simple feature set is that typically not every participant is interested in every part of the meeting [5]. Consequently, different sets of participants are more likely than the others to speak during the discussion of different topics during a meeting. Thus by tracking the speech lengths of the participants, we hope to detect transitions between topics. This feature is also used by [4] as input to their boundary classifier.

2.2. All–Words Based Boundary Prediction

The all–words based boundary predictor also uses a TextTiling based approach to output topic boundary predictions. The features used to compute similarity between adjacent windows are based on the words spoken by the various meeting participants during those two windows. These words can be output by an automatic speech recognizer; for the results in this paper however, we use manual transcriptions.

First we create the *vocabulary* of the meeting by determining the set of all the words spoken by all the participants in the entire meeting. We prune this set by removing *stop* words that typically contain low information (such as the articles, prepositions, etc.), and by removing morphological inflections from the remaining words using the Porter stemmer. Next, given a particular window, we create a feature vector with as many dimensions as the size of the meeting vocabulary. Each dimension in this vector corresponds to a different word in the vocabulary, and represents the number of times that word (or morphological inflections of that word) was spoken in that window. Note that although inflection removal introduces some noise into the feature vectors (since it is non–trivial to determine the correct root of some inflected words such as *axes* (*axe* or *axis*?)), our empirical evaluations show that stemming results in improved performance. The motivation for this boundary predictor is the same as that of the original TextTiling work [1] – that if two windows are on the same topic they will have a large number of overlapping words.

2.3. Combining Boundary Predictions

For a given meeting, we run both the speech activity as well as the all–words based boundary predictors. This results in two sets of similarity values for each time point in the meeting. We combine these values by computing for each time point a linear weighted combination of the similarity values from the two algorithms at that time point. That is, for each time point t, we compute its final similarity value $sim(t)$ is computed as:
$sim(t) = w \times speechActivity(t) + (1 - w) \times allWords(t)$
where $speechActivity(t)$ is the similarity value computed by the speech activity based boundary predictor at time point t and $allWords(t)$ is the similarity value computed by the all–words based boundary predictor at time point t. Weight w is a value less than 1, and is trained from a separate development data set. Given these combined similarity values, we then calculate depth scores, and report boundary predictions as described above.

3. Evaluation

3.1. The Annotated Meeting Corpus

Meeting data is being collected as a part of the CALO project. A subset of this data collected in 2003 and 2004 at Carnegie Mellon University and SRI International is "scenario driven". A group of 3 or 4 participants are given a broad topic to discuss over a period of 5 meetings. The topic involves hiring personnel, and acquiring new hardware and office space for them. Before each meeting, participants were asked to decide on an agenda for the meeting, and then adhere to the agenda during the meeting. Afterwards, a human annotator marked the time points in the meetings where the participants stopped speaking about an agenda item, and moved to the next one. Although human annotators typically do not achieve high agreement when identifying topic shifts, our annotation task is easier since we already have the agenda from each meeting, participants generally adhere to the agenda, and our goal is to simply detect when the discussion moves from one agenda item to the next.

Although these meetings were scenario driven, there was no fixed script for the meetings. Participants were given broad control over the evolution of the "back–story" from meeting to meeting. The data we use in this paper consists of 3 sequences of meetings named CMU-2, CMU-3 and SRI-1. Of these the first sequence had 4 meetings, and the other two had 5 each, for a total of 14 meetings. There was no overlap of participants from one sequence to another. On average each meeting was 15 minutes long, and had 5 agenda items each.

3.2. Evaluation Methodology

We evaluate our topic boundary prediction algorithms by comparing the automatically generated boundaries to the manually annotated topic boundaries in the test data. Our approach consists of first counting the number of automatically detected boundaries that *match* one or more manually annotated topic boundary, and then computing *precision*, *recall* and *f–measure* based on the number of matches. An automatically detected boundary is defined to match a manually annotated boundary if they occur within n seconds of each other, where n is a parameter that can be modified to obtain different precision/recall tradeoffs. Based on the number of matches, we compute precision as the ratio of the number of matched boundaries to the number of boundaries the algorithm reports. We compute recall as the ratio of the number of matched boundaries to the number of annotated boundaries. We compute f–measure as the harmonic mean of the precision and recall, with equal weight on both values. This evaluation methodology is very similar to evaluation of various different language technologies such as document retrieval, word sense disambiguation, etc. In addition to reporting the precision/recall/f–measure values, we also report the average absolute difference in time between the matched boundaries. This number is always $\leq n$ – the parameter that defines how far apart in time the automatic and the manual boundary can be to still qualify as a match.

3.3. Training, Development and Test Regime

There are two parameters in our algorithm we train – the appropriate window size for the two boundary predictors, and the best weight with which to linearly combine them. [We do not train the minimum topic length parameter, and simply set it to 60 seconds for these experiments]. To perform one round of cross validation, we assign one of the three meeting groups as the training set, another as the development set and the third as the test set. This gives us 6 possible permutations of assignments.

Given such an assignment, we first optimize the window sizes of the two boundary predictors. We use each predictor to detect boundaries on the training data using window sizes from 10 to 240 seconds at 10 second intervals. We compute f–measure at each window size for both predictors, and pick the window size at which we get the highest f–measure. Next we use the development set to find the optimum weight with which to linearly combine the similarity values from the two boundary predictors. We iterate through weights from 0.0 to 1.0, at intervals of 0.1. At each weight iteration we run each of the two boundary prediction algorithms on the development data using their optimum window sizes trained on the previous step, and combine the similarity values from the two predictors using the weight at that iteration. We compute f–measure at each step, and pick the weight with the highest f–measure. Finally, we run the algorithm on the test data using the trained window sizes and combination weight, and compute precision, recall and f–measure.

On performing this experiment across all the 6 data combinations, we get an average precision of 0.85, recall of 0.59 and f–measure of 0.67 on unseen test data. The high precision but low recall values imply that typically the boundaries detected by the algorithm are close to the real boundaries, but that not enough boundaries are reported. This can be modified by changing the threshold of acceptance of candidate boundaries. The exact "best" window size for the two predictors changes based on the training data; the average being 56 seconds for the speech activity predictor and 70 seconds for the all words predictor. The average best weight for linear combination is 0.6 for the speech activity boundary predictor (and 0.4 for the all words predictor).

4. Error Analysis

To gain a better understanding of which aspects of the topic detection algorithm perform well and which are problematic, we performed a manual analysis of the test results presented above. To do so, we first plotted the similarity values and the resulting depth scores from the two boundary predictors on a randomly chosen meeting. Figure 1 represents the similarity and depth score plots of the speech activity boundary predictor, and figure 2 those of the all words based boundary predictor on the same randomly chosen meeting. Both figures show three plots. The solid vertical lines represent the times in the meeting that the human annotator has classified as a topic boundary. For the chosen meeting the annotated topic boundaries are at 274, 432, 479, 599 and 652 seconds from the start of the meeting. The dotted curve shows the similarity values at each time point in the meeting, while the dotted vertical lines show the depth scores computed at each valley point. Similarity values range from 0 to 1, and depth scores from 0 to 2. In the remainder of this section we will refer to time points at t seconds from the start of the meeting as "time point t".

4.1. Analyzing Speech Activity based Boundary Predictions

Observe from figure 1, that the boundaries at time points 479 and 599 have high depth score valleys within an interval of 30 seconds (at time points 495 and 596 respectively), while the boundary at time point 432 also has a valley (albeit with lower depth score) nearby at time point 429. The boundaries at time points 274 and 652 however have no high depth score valleys nearby at all. Lis-

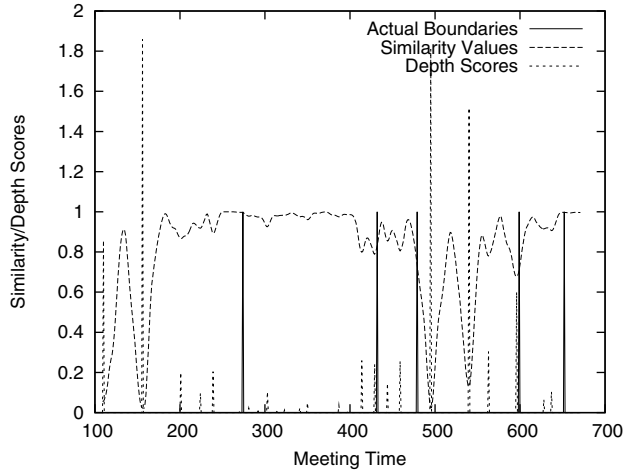

Figure 1: *Similarity and Depth Score Plot for Speech Activity based Boundary Detection for a Sample Meeting*

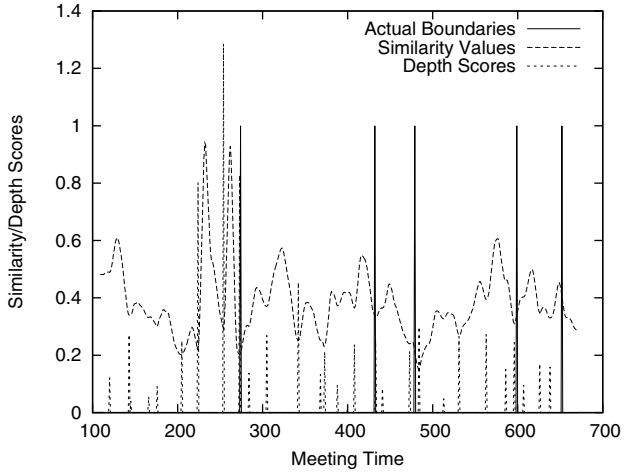

Figure 2: *Similarity and Depth Score Plot for All Words based Boundary Detection for a Sample Meeting*

tening to the audio of the meeting around those two time points reveals that the meeting facilitator verbally concluded the previous agenda item and then introduced the next item at that time point, without any interruption from the other participants. Hence the windows on the two sides of those time points are highly similar. A larger window size may help the algorithm, since the other speakers do eventually speak in both cases. Equally problematic are the high depth scores at time points 110, 156 and 540 none of which are close to a manually annotated boundaries. Those deep valleys are caused because a certain participant starts speaking at time point 110 and remains the sole speaker until till time point 156. Such behavior goes against the speech activity based boundary predictor's naive assumption that different topics have different degrees of participant involvement, and that their speech contributions are uniform through the course of the topic. A similar phenomenon happens at time point 540.

4.2. Analyzing All–Words based Boundary Predictions

Unlike the speech activity boundary detector, the all words boundary detector has many more valleys, several with high depth scores, as shown in figure 2. In fact the boundaries at time points 274, 432, 479 and 599 all have valleys with high depth scores close by (at 273, 434, 484 and 596 respectively), and the valley at time point 638 may be considered "close enough" to the boundary at time point 652. Unfortunately there are many other valleys with high depth scores that are not close to boundaries. For example the high depth score at time point 224 is caused due to the fact that the participants were speaking about buying computers just before that point, but then switched to talking about when the computers need to arrive. Although these two discussions segments are both part of the same agenda item, they have relatively low word overlap. These can be considered as finer grain topic shifts, that the all words algorithm is particularly suited to detect.

5. Conclusions and Future Work

In this paper we have presented a TextTiling based algorithm to detect topic boundaries in meetings. Our algorithm achieves a precision of 0.85 and recall of 0.59 on unseen meetings. Our error analysis suggests that our algorithm is not robust to atypical participant behavior. Our future plans include experimenting with other sources of information (e.g. prosody and cue–phrases), as well as other algorithms for boundary detection besides TextTiling. Additionally, we plan to work on techniques to automatically improve detection performance over a sequence of related meetings.

6. Acknowledgements

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.

7. References

[1] M. Hearst, "TextTiling: Segmenting text into multi–paragraph subtopic passages," *Computational Linguistics*, vol. 23, no. 1, pp. 33–64, 1997.

[2] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation," *Machine Learning*, vol. 34, no. 1-3, pp. 177 – 210, 1999.

[3] 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*, 2004, pp. 113–120.

[4] 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*, Sapporo, Japan, 2003, vol. 1, pp. 562 – 569.

[5] 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.