Linear Discourse Segmentation of Multi-Party Meetings Based on Local and Global Information

Mohammad Hadi Bokaei, Hossein Sameti, and Yang Liu, *Senior Member, IEEE*

***Abstract*—Linear segmentation of a meeting conversation is beneficial as a stand-alone system (to organize a meeting and make it easier to access) or as a preprocessing step for many other meeting related tasks. Such segmentation can be done according to two different criteria: *topic* in which a meeting is segmented according to the different items in its agenda, and *function* in which the segmentation is done according to the meeting's different events (like *discussion*, *monologue*). In this article we concentrate on the function segmentation task and propose new unsupervised methods to segment a meeting into functionally coherent parts. The first proposed method assigns a score to each possible boundary according to its local information and then selects the best ones. The second method uses a dynamic programming approach to find the global best segmentation according to a defined cost function. Since these two methods are complementary of each other, we propose the third method as a combination of the first two ones, which takes advantage of both to improve the final segmentation. In order to evaluate our proposed methods, a subset of a standard meeting dataset (AMI) is manually annotated and used as the test set. Results show that our proposed methods perform significantly better than the previous unsupervised approach according to different evaluation metrics.**

***Index Terms*—Dynamic programming approach, linear discourse segmentation, meeting function segmentation, unsupervised algorithm.**

I. Introduction

MEETINGS are important activities for businesses, managements and organizations in where people spend plenty of their daily time. Organizing these data is traditionally done by human operators. The growing number of daily meetings makes it very expensive or impossible to maintain and structure all the information exchanged in them. Accordingly automatically understanding and structuring meetings has attracted a great deal of attention in the last decade. However, understanding a meeting thoroughly (in a way that all the related questions can be answered and the minutes can be extracted automatically) is a challenging task for computers, and hasn't been achieved yet. Instead several research groups have tried to analyze the meeting data according to different aspects such as action item detection [1]–[3], decision summarization [4], [5], sentiment and opinion analysis [6]–[9], and discourse segmentation [10]–[18]. All these tasks extract specific information from the meeting data according to a predefined goal.

Among these attempts, linear discourse segmentation aims to segment a meeting transcript into a linear sequence of separate parts. This segmentation can be done according to different criteria. One of the most attractive trend is topic segmentation in which the main goal is to segment a meeting transcript according to items in the meeting agenda. This type of segmentation has been studied for many years and various algorithms have been proposed in the literature [10], [15]–[17]. However there is another natural way to partition a meeting transcript: it can be segmented into separate activities such as *discussions* between all or some of the participants, *monologues* or *presentations*. This kind of segmentation is called function segmentation [19]. In each segment, a specific set of participants are involved in the progress of the segment, which are called *active speakers*. Each of these participants has a function role in the segment, such as presenter, inquirer, responder, discussant, etc.

In this work we focus on the function segmentation of the meetings, which has not been studied well in previous work. The term *function* is originally used to refer to the *purpose* of the communicative acts. Similarly in our proposed segmentation schema, the main purpose in each segment differs from the ones in the adjacent segments. Two examples from a meeting in the AMI corpus[1] are shown in Fig. 1. In Fig. 1(a) a monologue segment is shown in which the purpose is to illustrate a specific issue for the attendance. Participant B is a presenter and lectures all other participants. Fig. 1(b) is an example of a discussion segment in which the purpose is to clarify a raised question. Participant A is inquirer and participants B and C are responders. In this work we just focus on the segmentation part and we leave the role identification task as our future work.

In our previous work [18], we showed that topic-based segmentation is not as effective as function-based counterpart for extractive summarization algorithms. Those results motivate us to further study the function segmentation task and propose more accurate algorithms. It must be noted that this type of segmentation extracts different piece of information from the meeting transcript and structures it in a different way from that in the topic segmentation. Obviously the combination of these two segmentations is possible, however it depends on the granularity of the topic definition. If we use coarse-grained topics where a meeting is segmented into high level goals (such as different and unrelated problems to be solved in a single meeting),

Manuscript received November 22, 2014; revised April 06, 2015; accepted June 29, 2015. Date of publication July 14, 2015; date of current version July 22, 2015. This work was supported in part by the National Science Foundation (NSF) under award IIS-0845484. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Murat Saraclar.

M. H. Bokaei and H. Sameti are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (e-mail: bokaei@ce.sharif.edu).

Y. Liu is with the Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080 USA.

Digital Object Identifier 10.1109/TASLP.2015.2456430

[1]AMI corpus will be explained later in Section III-A.

B: i only have three slides, so.
B: i just look at the mm -lcb- disfmarker -rcb- um just this. on some web pages to find some documentation
A: yeah .
B: and i think a remote control is, as i s mentioned previously,
B: you just have a a very simple chip and the mm the user interface is just done usually by push button
B: and in our case we are using a um a wheel control.
B: so uh uh i was looking basically for that chip, which is uh very very standard, and uh i just looked for the wheel sensor and the standard push button.
B: and um -lcb- vocalsound -rcb-
B: yeah we can change directly.
A: yeah.
B: in fact i have the number of that element which is very standard for remote control.
….

(a)

C: mm-hmm.
C: i have a question about that actually -lcb- vocalsound -rcb-.
C: um, what is the purpose of the light?
B: just to to make something which is uh slightly more design that uh a squarey box with a rubber -lcb- disfmarker -rcb-
C: is -lcb- disfmarker -rcb-
A: you can easily find the button in the dark or so?
C: but -lcb- disfmarker -rcb-
C: but in th in the dark uh -lcb- disfmarker -rcb-
C: yeah but is going to be always turned on , the light?
A: it will be turned on when the when the user move the remote control i think,
A: no?
C: but if you move it then you have it , you do n't need to find it.
A: hmm.
C: you can see the buttons better , of course.
…

(b)

Fig. 1. Examples of two function segments from the AMI corpus.

then each topic segment can be further segmented according to function criterion. In this way a hierarchical structure can be extracted from the meeting transcript in which the first level is the high-level problems and the second level is the function segments in each problem.

One of the earliest work on function segmentation in the meeting domain proposed an HMM-based approach to recognize group actions in a given meeting transcript [11]. The defined group action types in this work are: *discussion* where most participants engage in a conversation, *monologue* where one participant speaks continuously without interruption, *note-taking* where most participants take notes, *presentation* where one participant presents using projector screens, *white board* where one participant speaks using the white board, and *consensus* and *disagreement* where all participants express consensus or disagreement respectively. They also considered three combined categories in their work: monologue/presentation/white-board with note-taking. In order to train the model, they collected a simple corpus, called M4, which contains very short artificial meetings with their corresponding segmentations. Since then, three separate groups have continued to work on this task using the same corpus, with a more or less similar category set: IDIAP, Edinburgh, and Technical University of Munich.

IDIAP tested various kinds of Hidden Markov Models in order to detect the sequence of group actions of a given meeting [12], [20]–[23]. In [20] a set of *individual actions* was defined describing the act of participants in each time slice. These actions were *speaking*, *writing*, and *idle*. They introduced a two-layer HMM in which in the first step these individual actions were recognized, and then in the second step group actions were determined. The authors in [22] defined several streams of features (audio/visual stream for each participant) and then evaluated various types of HMMs to combine these streams: Simple HMM, Multi-stream HMM [24], Coupled HMM [25], and Asynchronous HMM [26]. A semi-supervised approach was also introduced in [23] in which a set of unlabeled data were used to improve the performance of the HMM.

Edinburgh group tried the Dynamic Bayesian Network approach and considered various kinds of features in their work [13], [27], [28]. In these works the segment categories were decreased to five possible actions: monologue, discussion, note taking, presentation, and white-board. They used the same M4 corpus and introduced various prosodic, lexical, and video features to train their DBN models. They compared their results with a simple HMM approach and reported good improvement over that.

Another group in Technical University of Munich tested several classifiers to segment a meeting [14], [29], [30]. They used the same segment categories as defined in studies by Edinburgh. The authors in [29] trained a classifier to detect the type of a single segment and then used a similar approach to the BIC algorithm [31] to find the segmentation. Specifically two connected windows were rolled over the input sequence and the boundary was detected wherever the predicted categories for those windows didn't agree. They also reported the results of a combination of neural network and HMM model in [30].

A comparative study of these approaches can be found in [32] in which different feature sets and models defined in these three groups were compared. It was shown that the best results were achieved when the multi-stream DBN model [13] was used.

We believe that these approaches have specific drawbacks as following:

1) *Task definition:* The group actions were not well defined. They didn't discriminate between various kinds of discussions. In fact in a real meeting it is probable that two consecutive discussion segments have different participants engaged. These previous approaches considered all these consecutive segments as one long *discussion* segment. We believe that it will be useful to discriminate between those different segments. For example in a summarization task, it would be good to have these segments separated, then we can use the active speakers' information in each segment

to extract the most important utterances. As the simplest case, we can simply omit the utterances of the participants other than active speakers and then select the most important utterances from the remained ones.

2) *Supervised models:* They were all supervised and needed a training set to adjust the parameters of their models. Obtaining such data is a time-consuming, expensive and error-prone process.
3) *Data:* They all evaluated their results on the M4 corpus [11] which contains artificial short meetings. The meetings in M4 were collected in a scripted way where a sequence of group actions with their durations had been first generated and then a group of participants discussed according to these generated scripts. On average each meeting contains 5 actions and the total time for the whole meeting is approximately 5 minutes. A dedicated timekeeper monitored the ongoing action duration and made silent gestures to prompt transition to the next action in the script. These artificial transitions along with the shortness of each meeting make this corpus to be far from a real one.

There is also other research conducted outside these three groups [33], [34] and they more or less suffer from similar problems. The only unsupervised algorithm was introduced in [35] in which the authors proposed an algorithm to detect dynamics of group configurations in various types of conversations. They rolled two fixed-length windows over the observation sequence and compared the histogram of various speaker combinations in adjacent windows. This procedure was repeated for different values of window lengths and boundaries were detected in points on which the majority of those repetitions agreed. The overall structure of this algorithm is standard and mostly used in speaker diarization approaches [36]. However they just considered discussions between two participants. There is no monologue nor discussion between more than two participants in their target category set. Their approach also had many parameters on which the performance of the algorithm is highly dependent.

In this article we concentrate on segmentation methods which address all the mentioned drawbacks of previous work:

1) *Task definition:* In this work, a new set of categories is defined wherein various kinds of discussions are discriminated according to their involved participants.
2) *Unsupervised:* We concentrate on unsupervised approaches with no need to training data. Three new unsupervised methods are proposed in this work: the first one uses the local information. It tracks the change in the flow of the meeting and decides about the boundaries on the points where this change is most remarkable. The second method uses a dynamic programming approach to find the global best segmentation according to a defined cost function. Each one of the two methods has its own advantages. We finally combine these approaches and come up with the third method which takes advantage of the first two methods and improves their results.
3) *Data:* In order to evaluate our proposed algorithms on a more realistic dataset, a subset of the AMI corpus [37] was manually annotated. All the experimental results are reported according to this annotated test set.

Another related work was proposed in our previous work [18] in which a genetic based algorithm was introduced for this task. We defined a scoring schema for segmentation and then optimized it using genetic algorithm to find the best function segmentation. The main problem of this work is its dependency on the initial population, which leads to different results and makes the algorithm to be unstable.

Since the main usage of the segmentation algorithms is as the preliminary step in other meeting understanding tasks like summarization, we assume that the transcript of the meeting is available using an automatic speech recognition engine. Unlike previous work, observation units in our work are language units (words or sentences) instead of time unit.

The rest of the paper is organized as follows. In Section II our proposed algorithms are explained. We first introduce our basic approaches which use local and global information to segment a meeting. We then discuss the benefits of these basic approaches and explain the idea leading to the combined algorithm. In Section III we introduce our test set and the experimental setup. We then report the performance of the proposed algorithms in various conditions and compare them with each other as well as other base-line approaches. Finally in Section IV we conclude the paper and discuss our future works.

II. Proposed Methods

Here we define a new set of target categories to segment a meeting accordingly: $Discussion_{x_1 \ldots x_n}$ and $Monologue_{x_1 \ldots x_n}$, where x_i is a binary value which denotes whether speaker i is active in the segment or not. For example, $Discussion_{1010}$ denotes a segment where participants 1 and 3 have a discussion, or $Monologue_{0001}$ denotes a segment where participant 4 lectures all other participants. Note that the number of categories changes depending on the number of participants. Generally for a meeting with N_s participants, we have $2^{N_s} - 1$ potentially different categories. For example, in a meeting with four participants we have 15 categories.

The input to our segmentation algorithms is the sequence of unit elements. In this work we examine both word and sentence to be the unit element constructing the input sequence. As we have argued above, since the main usage of this algorithm can be considered as the first step of other meeting understanding tasks, the transcript and the speaker of each sentence/word are assumed to be prepared using an automatic speech recognition engine.

We first introduce local and global segmentation methods and then discuss their advantages and disadvantages. We then introduce our proposed way to combine these approaches in a single method to improve the results. All these algorithms are applicable regardless of the type of the input, word or sentence sequence.

A. Local-Based Segmentation: Statistic-Based Approach

The main idea in this method is to track the change in the input sequence and determine boundaries wherever this change is maximized. The general structure of this method is similar to the Texttiling algorithm [38], which is originally used to find the topic boundaries within a document. The main idea in the

Texttiling algorithm is that the vocabulary does not revolve dramatically during the course of one topic segment, however it undergoes a significant change wherever the topic is shifted. In this approach two windows are rolled over the elements in the input word sequence and compared according to the patterns of lexical distribution.

In this article, however, we are interested in the function segmentation of a meeting transcript. For this purpose, we can't simply use the lexical patterns to differentiate these function segments from each other, since it is probable that we have two adjacent function segments with similar topics or one function segment consisting of different topics. As a result, significant change in the lexical pattern is not a good indicator for shifting in a function segment. Accordingly we propose a new approach to compare two adjacent windows.

The main idea in our approach is that the involved participants set is not changed significantly during the course of one function segment. In contrast there must be a noticeable change in this set between two segments. We try to find these points in the input sequence.

All the elements in the input sequence are considered to be a possible boundary for the segmentation. For each possible boundary at position i, two windows are placed over elements $[i-L_w : i-1](win_{i,left})$ and $[i : i+L_w-1](win_{i,right})$. L_w is the window length and will be tuned according to our development set. From each window, two sets of features are extracted to be used to find the change points.

As the first group of features, the contribution of each participant is measured in terms of its uttered words count. $WC_j(win)$ corresponds to the jth participant in the window win and is calculated according to Equation (1).

$$WC_j(win) = \frac{num\_word(j, win)}{\sum_k num\_word(k, win)} \tag{1}$$

where $num\_word(j, win)$ is the total number of words uttered by participant j in the window win.

For the second group of features, the importance of words uttered by each speaker is measured. There are various weighting metrics used in the literature. We choose $suidf$ [39] to measure the importance of each word. In most of the conventional metrics (like $tfidf$ [40]), words are scored according to their distribution across a collection of documents. Words which are common in a single document but rare in other documents gain a higher score. However in $suidf$ formulation, the variation of word's usage among participants is also considered. The idea in this metric is that important words are not used by all the participants with the same frequency. It is previously shown that this metric reflects the importance of a word better than the conventional ones for summarizing multi-party conversations [39]. In order to calculate this metric, we first describe the surprisal score of participant s utters word w:

$$surp(s, w) = -\log\left(\frac{\sum_{s' \neq s} tf(w, s')}{\sum_{s' \neq s} N(s')}\right) \tag{2}$$

where $tf(w, s)$ is the total number of times that speaker s utters the word w and $N(r)$ is the total number of words uttered by speaker r. Generally speaking $surp(s, w)$ measures how much surprising it is if the word w is uttered by participant s. If the word w is commonly used by speakers other than s, then this word is not a special word and the score $surp(s, w)$ will be a low value. On the other hand, if the word w is rare among other speakers, then its usage by s is surprising and, consequently, the score $surp(s, w)$ will be a high value.

The surprisal score of the word w, will be the average of $surp(s, w)$ over all participants:

$$surp(w) = \frac{1}{N_s}\sum_s surp(s, w) \tag{3}$$

where N_s is the total number of participants in the meeting. $surp(w)$ measures how informative a word is based on how it is used by different speakers.

Finally the metric $suidf$ for the word w is calculated according to Equation (4):

$$suidf(w) = surp(w).\frac{s(w)}{N_s}.\sqrt{idf(w)} \tag{4}$$

where $s(w)$ is the number of participants who utter that word and N_s is the total number of participants in the meeting. $idf(w)$ is the conventional *inverse-document-frequency*. To calculate idf score in this work, each utterance is counted as one single document. Using $s(w)$ in the formula gives higher weight to words that are used by more participants. The influence of the idf score is also reduced by using the square root (which was determined empirically in [39]).

The second group of features is also a vector with length N_s in which the jth element (corresponds to the jth participant) is calculated according to Equation (5).

$$WI_j(win) = \frac{\sum_{w \in Words_j^{win}} suidf(w)}{\sum_k \sum_{w \in Words_k^{win}} suidf(w)} \tag{5}$$

where $Words_j^{win}$ is the total words which are uttered by participant j in the window win.

The score of the possible boundary i is considered to be the distance between the feature vectors extracted from $win_{i,left}$ and $win_{i,right}$ according to Jeffrey divergence [41], which is a numerically stable and symmetric form of the Kullback-Leibler distance metric [42]:

$$\begin{aligned} score(i) = {} & dist_{jef}(WC(win_{i,left}), WC(win_{i,right})) \\ & + dist_{jef}(WI(win_{i,left}), WI(win_{i,right})) \end{aligned} \tag{6}$$

where

$$dist_{jef}(p, q) = \sum_i p(i).log\frac{p(i)}{\frac{p(i)+q(i)}{2}} + q(i).log\frac{q(i)}{\frac{p(i)+q(i)}{2}} \tag{7}$$

where $p(i)$ and $q(i)$ are the ith elements of the vectors p and q respectively.

These scores are computed for each possible boundary to form the score plot. This process leads to a rough plot with a

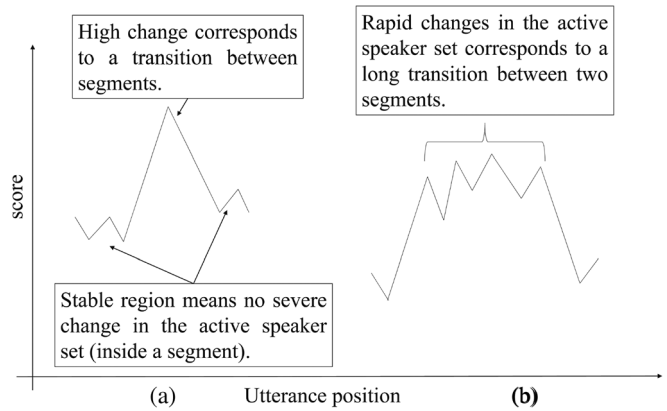


Fig. 2. Illustration of the usefulness of the $peak_{score}$ formulation.

lot of small up and downs. Therefore it is necessary to smooth it. We use the following smoothing function:

$$score(i) = \frac{1}{s+1} \sum_{j=i-s/2}^{i+s/2} score(j) \tag{8}$$

where s is the parameter of the smoothing algorithm. The $peak\_score$ is then obtained for each position according to Equation (9):

$$peak\_score(cp) = 2 * score(cp) - score(pw) - score(nw)\& \tag{9}$$

where cp is the point at which we want to calculate the $peak\_score$, and pw and nw are the locations of the nearest valleys (local minimums) to cp on its left and right sides respectively.

Note that the $peak\_score(cp)$ measures the relative strength of the change at the point *cp* in compare with its adjacent scores. The reason behind using this relative schema rather than simply using the absolute score (Equation (8)) is that we want to select points with high changes in active speaker set (results in high peaks in the score plot) that have a stable region before and after them (results in deep valleys in the score plot). An example is shown in Fig. 2(a). Another benefit of using relative schema is that we want to prevent selecting adjacent local maximums which are in the same range and there is no deep valleys between them (Fig. 2(b)). This situation can occur in a transition between two segments where there is a high fluctuation in the active speaker set. We don't like to return all these adjacent local maximums as the segment boundaries, since they all correspond to one segment transition.

Finally we return the boundaries with the highest $peak_score$s. If the number of segments is known in advance, we simply return the top best scored boundaries. In the case that this number is not known (the more realistic case), we calculate the average ($\bar{s}$) and the standard deviation (σ) of all the calculated $peak_score$s and return candidate boundaries whose $peak_score$s are higher than $\bar{s} - \sigma$. The complete algorithm is shown in Fig. 3.

Input: seq (Input sequence with length L)
Parameters: L_w (window length), s (smoothing parameter)
for $i = L_w + 1$ to $L - L_w$ **do**
  Calculate $score(i)$ according to Equations 6, 7;
end for
Smooth scores according to Equation 8;
Calculate $peak\_score(i)$ according to Equation 9;
$\bar{s} = average(peak\_score)$;
$\sigma = standard\_deviation(peak\_score)$;
$boundaries = \{i | peak\_score(i) > \bar{s} - \sigma\}$;
Output: $boundaries$.

Fig. 3. Local algorithm to find the function segmentation of the input sequence.

B. Global-Based Segmentation: Dynamic Programming Approach

The main idea in this approach is to define a function to score a hypothesized segmentation and then use a dynamic programming algorithm to find the global best segmentation according to that defined function. The input sequence to this algorithm can be word/sentence sequence. However it is more appropriate to apply this algorithm on the sentence sequence, due to its high computational cost, which depends on the input sequence's length. This issue is more explained in the next subsection.

For this algorithm, we assume that the number of segments is known or estimated in advance and stored in Num_{seg}. For the case that this number is unknown, we use the same number of segments as is found in the previous statistic-based approach.

We first introduce some definitions used in our algorithm. For each segment s, we define its speakers' distribution $dstr(s)$, which is a vector with elements denoting the fraction of words expressed by each speaker in that segment[2]. We also define a unique distribution for each target category, in which all the corresponding active participants are equally involved (having the same number of words). This is referred to $dstr_{ideal}(N_s, cat)$ where cat is the corresponding category and N_s is the number of participants in the meeting. Fig. 4 shows a sample sequence s, its distribution ($dstr(s)$) and $dstr_{ideal}$ for two sample categories, namely $Monologue_{0100}$ and $Discussion_{1010}$.

The idea behind our proposed algorithm is to create a segmentation consisting of segments whose distributions are more similar to the ideal distributions. The score of a segmentation is considered to be the sum of its segments' scores. In order to calculate the score for a segment, we divide the segment into smaller parts, each with length M. In this work, we set M to be twice the number of participants in the meeting. The score for a sample segment s is:

$$score_{seg}(s) = \frac{1}{L-M} \min_{cat \in C} \sum_{i=1}^{L-M} \{dist_{jef}(dstr(s(i:i+M)), dstr_{ideal} N_s, cat)\}) \tag{10}$$

where L is the number of elements in the input segment s and N_s is the number of participants in the meeting. $s(i : i + M)$ is the sub-segment of s with length M starting from index i. C

[2]This is similar to the definition of WC vector in the statistic-based approach.

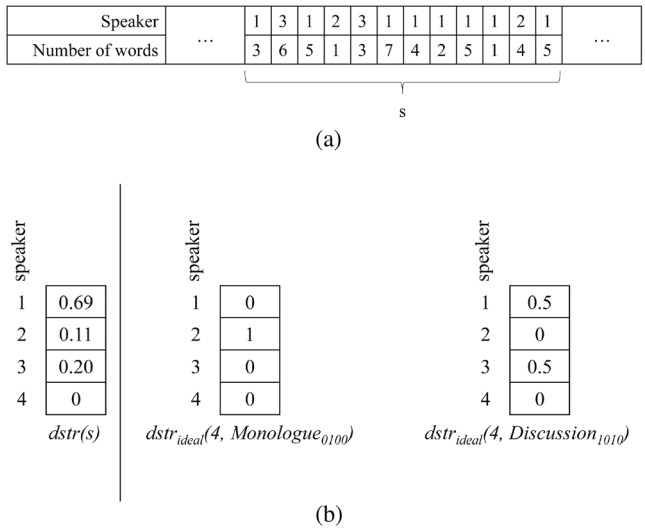


Fig. 4. Example of $dstr$, $dstr_{ideal}$ for a sample segment in a meeting with 4 participants. (a) The sample segment, s, in the input sequence. The first row is the speaker of the sentence and the second row is the number of the words in the sentence. (b) The distribution of the segment and two ideal distributions according to two different categories.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
(a)	3	2	3	3	2	3	1	3	1	3	1	1	3	3	3	2	2	2	2	2
(b)	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0

Fig. 5. (a) Sample sequence of speaker id's for a fictitious meeting with 3 participants and 20 utterances. (b) Sample segmentation of the given sequence. This segmentation contains three separate segments which start at positions 1, 7 and 16.

is the list of categories defined at the beginning of this section. $dist_{jef}(P, Q)$ is the Jeffrey distance metric which is shown in Equation (7).

In our defined scoring schema, the more similar a segment's distribution is to an ideal distribution, the lower its score is. As mentioned before, we like to find a segmentation whose segments are similar to the ideal distributions. For example in a monologue segment of type $Monologue_{1000}$, we like to have a segment in which all sentences are uttered by the first participant. Although this is not the case in real scenarios, due to other participants' activities like backchannels, it is desired to find segments as similar as possible to this ideal distribution. An example is shown in Fig. 5 for a fictitious meeting. The sequence shown in Fig. 5(a) is segmented into three segments, s_1, s_2 and s_3 respectively. It can be seen that s_1 is similar to the ideal distribution of category $Discussion_{011}$ ($dstr_{ideal}(3, Discussion_{011})$). s_2 and s_3 are similar to $dstr_{ideal}(3, Discussion_{101})$ and $dstr_{ideal}(3, Monologue_{010})$ respectively.

The reason behind dividing a segment into subsegments in Equation (10) is to avoid having large segments that as a whole are similar to an ideal distribution, but sub-regions have different distributions (e.g., sequence $111\ldots1222\ldots2333\ldots3$ as one segment versus three separate segments).

In order to find the global best segmentation according to the defined scoring function, a dynamic programming algorithm can be used. Specifically we define $sb(i, q)$ to be the score of the best found segmentation for the first i elements of the given sequence using q segments. $score_{seg}(i : j)$ is defined to be the score of the subsequence i to j as one segment according to Equation (10).

Our proposed dynamic programming approach is composed of the following three steps:

1) *Initialization:* The initialization is done using Equation (11):

$$sb(i, 1) = score_{seg}(1 : i) \quad \forall i \leq L) \tag{11}$$

where L is the number of elements in the given sequence.

2) *recursion:* The recursive formula is defined as Equation (12):

$$\begin{aligned} sb(i, q) &= \min_{j<i}\{sb(j, q-1) + score_{seg}(j+1 : i)\} \\ bp^*(i, q) &= \arg\min_{j<i}\{sb(j, q-1) + score_{seg}(j+1 : i)\} \\ 1 \leq i \leq L &\quad 2 \leq q \leq Num_{seg} \end{aligned} \tag{12}$$

The best score at each position is calculated according to previously computed scores. The best j is also stored in bp^* matrix to backtrack to find the optimal path. Iteratively applying Equations (12), $sb(L, Num_{seg})$ will be reached, which is the score of the best found segmentation according to the defined scoring function.

3) *Backtracking:* Starting from $bp^*(L, Num_{seg})$, the global best combination of boundaries can be found according to the back pointers stored in bp^*.

As we mentioned earlier, we assume that the number of boundaries is known (or estimated) prior to calling the dynamic programming algorithm. Traditionally we could consider a maximum limit for the number of segments and then let the algorithm decide the optimal number of boundaries by itself, i.e., wherever $sb(L, q)$ is minimized for $1 \leq q \leq q_{max}$ where q_{max} is the maximum limit for the number of segments. However in our task the segmentation score is a monotonically decreasing function in terms of the number of segments. As this parameter is increased, the total score of the segmentation decreases. As the ultimate case, when the number of segments is equal to the number of the elements (each element is in a separate segment), the segmentation score becomes 0. Therefore we have assumed that the number of segments should be provided when the algorithm is called.

C. Analysis of the Local and Global Methods

Both methods explained in the previous subsections have their own advantages and disadvantages. We will analyze these approaches from two distinct viewpoints and show that their benefits can be considered to be complementary of each other:

1) *Local vs. global view:* The most important advantage of the dynamic programming approach is that it has a global view of the whole input sequence when it decides about the boundaries of the segmentation. On the other hand, the statistic-based approach has a local view and examines each possible boundary according to its local information around it. It then selects boundaries which maximize this local distance metric. Having a global view of the whole input sequence is a good characteristic of the dynamic programming approach that makes it able to find

TABLE I
SUMMARY OF THE MOST IMPORTANT ADVANTAGES AND DISADVANTAGES OF THE DYNAMIC PROGRAMMING AND STATISTIC-BASED APPROACHES

Algorithm	pros	cons
Dynamic programming	Global view of the whole sequence	Very slow $O(L^3)$
Statistic-based	Very fast $O(L)$	Local view at each candidate boundary

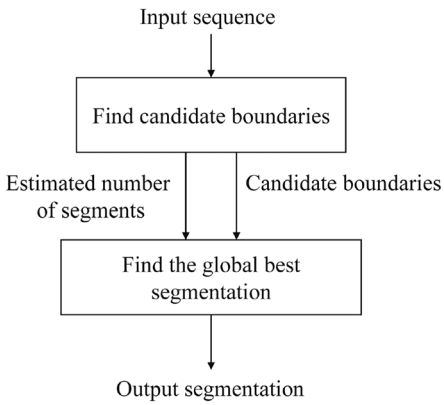


Fig. 6. The overall structure of the final proposed algorithm.

the best combination of boundaries according to the defined scoring mechanism.

2) *Computational cost:* The computational cost of the dynamic programming approach is much higher than the statistic-based one. Specifically the cost of the scoring function (Equation (10)) is $O(L)$ where L is the number of elements in the input sequence. In the dynamic programming approach, this segment score is computed for all subsegments j to i (Equation (12)) where $0 \leq i \leq j \leq L$. Therefor the total computational cost for our Dynamic programming approach is $O(L^3)$. The cost for the statistic-based approach is $O(L)$. In our task, L is large, especially when we apply the algorithm on the word sequence, resulting in a noticeable difference in the computational costs of these algorithms. This can be regarded as the most important advantage of statistic-based approach over the dynamic programming one.

These advantages and disadvantages are summarized in Table I. It can be seen that the advantages of these algorithms are complementary of each other. This is the main idea behind our new proposed approach which is explained in the next subsection.

D. Combined Algorithm

The main idea in this approach is to use a high-recall algorithm to find the candidate boundary set and then feed this set to the dynamic programming approach to find the global best segmentation. The overall structure of this algorithm is summarized in Fig. 6. As shown in this figure, the input word/sentence sequence is first analyzed to find the candidate positions that are more probable to be the segment boundaries. In this step, the number of segments is also estimated. This number as well as the candidate boundaries and the input sequence itself are passed to the second step in which the best combination of these candidates is found and returned as the output segmentation.

In order to find the candidate boundary set, we have to use an algorithm that has a low probability to miss a true boundary. In the meantime it has to eliminate non-boundary candidates as much as possible. Since we just want to find the candidate boundary set, there is no need to have a global view of the whole sequence. All the possible boundaries can be examined according to their local information and then boundaries that are detected to have a good potential to be in the true boundary set are returned. For this purpose we use a similar approach to the statistic-based algorithm (shown in Fig. 3). We use the same process to calculate the score for all possible boundaries, and obtain the score plot for the input sequence. For the candidate boundary set CB, we return all the local maximums of this plot:

$$CB = \{i | score(i) > \max(score(i-1), score(i+1))\} \quad (13)$$

where $score(i)$ is calculated according to Equation (6). In Section III, we will evaluate this algorithm in terms of how many true boundaries are missed in the output candidate boundary set. The only difference of this step with what we have explained in Section II-A, is that we don't compute the $peak_{score}$ anymore, since we don't care about returning redundant boundaries (like the ones shown in Fig. 2(b)). We also return all the local maximums as the candidate boundary set.

Next we use a global view to decide about the best combination among these candidate boundaries. For this step, we use a modified version of the dynamic programming approach to find the best segmentation. The only difference of this modified version of DP algorithm with the original one explained in Section II-B is the way that the elements in sb and bp^* tables are computed: here we just consider candidate boundaries which are extracted in the first step. This modified algorithm is shown in Fig. 7.

Since we apply the algorithm on the candidate boundary set, the computational cost of this new algorithm is $O(L_{cb}^2 * L)$ where L_{cb} is the number of found boundaries in the candidate boundary set. As shown in Section III, this number is much lower than L, which will be the number of elements in the input sequence.

As shown in Fig. 7, it is assumed that the number of the segments is known in advance before the dynamic programming method is called. Again we use the same number of segments as found in the statistic-based approach explained in Section II-A.

E. Labeling Problem

After segmenting a meeting transcript according to the function criteria, next we label each segment according to the defined target categories. To find this category for a segment s, we extract the speakers' distribution, $dstr(s)$ from that segment. We then utilize this distribution in two different ways:

1) *Mean-based*: In this approach, we calculate the mean of the elements in the $dstr(s)$ vector and then return participants whose corresponding element in the $dstr(s)$ is above the calculate mean.
2) *ideal-based*: In this approach, we compute the distance between the extracted $dstr(s)$ with all categories' ideal distributions according to Jeffrey divergence distance. We then

Input: seq (Input sequence with length L), CB (Candidate boundary set with size L_{cb}), Num_{seg} (Number of boundaries).
Initialization:
for $i = 1$ to L_{cb} **do**
  $sb(i,1) = score_{seg}(1 : CB_i)$;
  $bp^*(i,1) = 0$;
end for
Recursion:
for $i = 1$ to L_{cb} **do**
  for $k = 2$ to Num_{seg} **do**
    $sb(i,k) = \min_{j<i}\{sb(j,k-1) + score_{seg}(CB_j + 1 : CB_i)\}$;
    $bp^*(i,k) = \arg\min_{j<i}\{sb(j,k-1) + score_{seg}(CB_j + 1 : CB_i)\}$;
  end for
end for
Backtracking:
$n = Num_{seg}$;
$boundaries \leftarrow \{\}$;
$current \leftarrow bp^*(L_{cb}, n)$;
while $current > 0$ **do**
  Add $CB_{current}$ to $boundaries$;
  $n \leftarrow n - 1$;
  $current \leftarrow bp^*(current, n)$;
end while
Output: $boundaries$

Fig. 7. Evaluation process of a hypothesized segmentation according to a reference one.

identify the best matching category and return it as the label of that segment.

The performance of these approaches will be compared in the following section.

III. Experimental Results

A. Data

The AMI meeting corpus [37] is a collection of 100 hours of meeting data and includes annotations in various layers such as speech audio, transcripts, focus of attention, etc. These meetings are much more natural than the previously collected dataset for the meeting segmentation task (M4) [11].

In order to evaluate the effectiveness of our proposed algorithm, a subset of 11 meetings in this corpus are selected and manually annotated. The ids of annotated meetings are: *es2008a*, *is1000a*, *is1001b*, *is1001c*, *is1006b*, *is1007b*, *is1008a*, *is1008b*, *is1008c*, *ts3005a* and *is1003b*. We chose these meetings since they have more annotations (such as focus of attention and addressee information) in the AMI corpus, which can be useful for our future studies.

We employed graduate students as annotators and asked them to segment the meetings according to different events. They were given a guideline which included the task definition and various examples used to clarify the concept of events and function segmentation of a meeting. Each meeting was annotated by one annotator. One special trained annotator then checked and finalized the annotations. The average number of segments in this set is 19.63[3]. One meeting (*is1003b*) is used as the development set to tune the parameters of the algorithms, and the other ones are used as the test set.

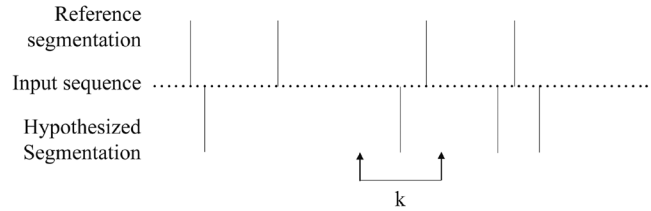


Fig. 8. Evaluation process of a hypothesized segmentation according to a reference one.

Automatic transcripts are provided by the AMI ASR team [43], yielding a word error rate (WER) of about 36%. We have evaluated our proposed algorithms according to two cases: in the first one we assume that sentence boundaries are known in advance. Recently proposed algorithms show good performance on automatically detecting these boundaries [44], [45]. In this case the elements of the input sequence to our algorithms are sentences. We use manually labeled sentences as provided in the AMI meeting corpus for each meeting. In the second case, we assume that there is no information about the sentence boundaries and we use the sequence of recognized words as the input sequence to our algorithms. In this work we just use the ASR transcription of the meetings as well as the speaker information of each sentence which are available in the AMI corpus. There are some additional symbols in the transcript (like vocalsound which indicates a sound like laugh, cough, etc) which are useless and are omitted before we feed the meeting transcript into the algorithms.

B. Evaluation Metrics

There are various segmentation evaluation metrics proposed in the literature. The most used one is P_k [46]. Given two points in a sequence, P_k specifies the probability of segmentation error, which is the average probability that the segmenter's decision is incorrect. The concept of the evaluation process for this metric is shown in Fig. 8. As shown in this figure, a window of length k is rolled over the input sequence. At each step, it is examined whether the hypothesized segmentation is correct about the separation of the two ends of the window or not. The formula is shown in Equation (14):

$$P_k = \frac{\sum_{i=1}^{L-k} \delta_H(i, i+k) \oplus \delta_R(i, i+k)}{L-k}) \qquad (14)$$

where L is the number of elements in the input sequence and H and R are the system generated and reference segmentation respectively. Given a segmentation S, $\delta_S(i,j)$ is a function which outputs 1 if and only if the segmentation S assigns ith and jth elements to the same segment. Note that P_k is a measure of error and thus a lower score means better segmentation performance.

It is possible to distinguish between *misses* where a true boundary is not detected and *false alarms* where a false boundary is detected by the algorithm. In [47] two metrics

[3]The reference segmentation as well as the annotation guide can be found here: http://ce.sharif.edu/bokaei/resources/funseg/

are defined which evaluate the performance of an algorithm according to its misses and false alarms separately:

$$P_{miss} = \frac{\sum_{i=1}^{L-k} \delta_H(i,i+k).(1-\delta_R(i,i+k)}{L-k}$$
$$P_{FalseAlarm} = \frac{\sum_{i=1}^{L-k}(1-\delta_H(i,i+k)).\delta_R(i,i+k)}{L-k}) \quad (15)$$

Although P_k is the most well-known metric for the segmentation task, it has several shortcomings. The authors in [48] pointed out these problems and instead introduced another metric called WindowDiff (WD) which works in a similar way to P_k by moving a window over the sequence. However in this metric, a window is scored as wrong if the number of boundaries within this window according to the system segmentation is different from that of reference segmentation. The formula is shown in Equation (16):

$$WD = \frac{\sum_{i=1}^{L-k}[|b_H(i,i+k) - b_R(i,i+k)| > 0]}{L-k}) \quad (16)$$

where $b_S(i,j)$ is the number of boundaries between i and j according to segmentation S. The WD metric can be decomposed into two parts:

$$WD = WD_{miss} + WD_{FalseAlarm}) \quad (17)$$

where

$$WD_{miss} = \frac{\sum_{i=1}^{L-k}[b_H(i,i+k) < b_R(i,i+k)]}{L-k}$$
$$WD_{FalseAlarm} = \frac{\sum_{i=1}^{L-k}[b_H(i,i+k) > b_R(i,i+k)]}{L-k} \quad (18)$$

The major drawback of the WD metric can be seen from Equation (18). Specifically the way that WD_{miss} is normalized is not fair, because the maximum number of misses depends on the number of boundaries in the reference segmentation, which is not reflected in the formula. Accordingly a new metric, called Pr, is defined in [49] which aims to fix this problem.

$$Pr = 0.5 * (Pr_{miss} + Pr_{FalseAlarm})) \quad (19)$$

where

$$Pr_{miss} = \frac{\sum_{i=1}^{L-k}[b_H(i,i+k) < b_R(i,i+k)]}{\sum_{i=1}^{L-k}[b_R(i,i+k) > 0]}$$
$$Pr_{FalseAlarm} = \frac{\sum_{i=1}^{L-k}[b_H(i,i+k) > b_R(i,i+k)]}{L-k}) \quad (20)$$

For the evaluation purpose in this article, we use P_k (the most popular one) and Pr (the most recent one) to compare the proposed algorithms.

In all the above metrics, the choice of k is arbitrary, but is generally set to be half of the average segment length in the reference segmentation. This value ensures that under some assumptions four obvious baseline algorithms (hypothesizing no boundaries, boundaries everywhere, evenly spaced boundaries, or randomly spaced boundaries) all have the expected value of 0.5 [50]. For our evaluation process, we use the same choice for k.

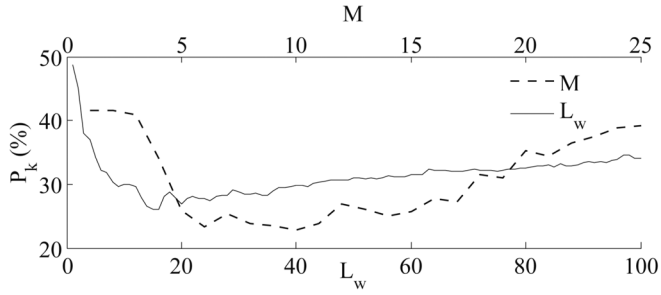


Fig. 9. The effects of the parameters of the statistic-based (solid line) and the dynamic programming (dashed line) algorithms on their performances evaluated on the whole test set.

C. Results

1) The Sensitivity of the Proposed Algorithms on Their Parameters: First we study the sensitivity of our proposed algorithms on their main parameters: window length, L_w, in the statistic-based and sub-segment length, M, in the dynamic programming algorithm. We report the performance of these algorithms on the test set, according to different values of their parameters in certain regions. The input to the algorithms is assumed to be the sentence sequence. The results are shown in Fig. 9. The solid line shows the performance of the statistic-based algorithm according to different values of L_w with respect to the bottom x-axis. Similarly the dashed line shows the sensitivity of the dynamic programing algorithm on its main parameter, M with respect to the top x-axis.

As can be seen from Fig. 9, these algorithms are not significantly sensitive to parameter values. As an example, changing the parameter of the statistic-based algorithm in the range [8, 60], the P_k value changes in the range [0.25,0.3]. The same result can be seen from the M plot. we can observe that the dynamic programming algorithm is not very sensitive to the parameter M. The only condition is that M must not be too small or too large. If it is very small, the comparison between the distribution of a sub-segment and its corresponding ideal distribution would be meaningless. If it is very large, the subdividing mechanism will be useless and the algorithm will be biased toward larger segments (refer to the reason behind subdividing the segment in our scoring schema in Section II-B). However this being large or small depends on the number of participants in the meeting. Accordingly the choice of M must depend on the number of participants. We empirically set M to be twice the number of participants in the meeting.

2) Comparison of the algorithms (Case 1: Known number of boundaries): First we assume that the number of boundaries is known in advance and we compare the performance of the algorithms accordingly. Results are shown in Table II. We compare the performance of our algorithm with the random segmentation and the previously unsupervised approach (Brdiczka *et al.* 2007) [35]. The Brdiczka approach is applied to time frames. However we apply our proposed algorithms on both sentence and word sequences. For both cases we use the ASR output as described in Section III-A. Because of the high computational cost of the dynamic programming approach, applying it to word sequence is not feasible. Accordingly for word level results, we just report the performance of the statistic-based and combined approaches.

TABLE II
PERFORMANCE OF THE PROPOSED ALGORITHMS IN COMPARISON WITH OTHER BASE-LINES. HERE WE ASSUME THAT THE NUMBER OF SEGMENTS IS KNOWN IN ADVANCE. THE NUMBERS IN PARENTHESIS ARE THE STANDARD DEVIATION

Input type	Algorithm	$P_k(\%)$	$Pr(\%)$
	Random	50.00	50.00
time frames	Brdiczka et al. 2007	36.31 (3.47)	39.52 (4.21)
sentences	Statistic-based	28.12 (6.90)	25.37 (5.93)
	Dynamic programing	28.21 (7.37)	25.52 (6.52)
	combined approach	26.06 (8.26)	23.89 (7.53)
words	Statistic-based	29.97 (6.15)	25.98 (5.66)
	combined approach	25.93 (7.66)	24.08 (6.71)

TABLE III
PERFORMANCE OF THE PROPOSED ALGORITHMS IN COMPARISON WITH OTHER BASE-LINES. THE NUMBER OF SEGMENTS IS AUTOMATICALLY ESTIMATED. THE NUMBERS IN PARENTHESIS ARE THE STANDARD DEVIATION

Input type	Algorithm	$P_k(\%)$	$Pr(\%)$
	Random	50.00	50.00
time frames	Brdiczka et al. 2007	38.92 (3.60)	43.99 (4.77)
sentences	Statistic-based	26.93 (6.06)	24.76 (5.84)
	Dynamic programing	28.13 (7.97)	31.97 (7.35)
	combined approach	26.03 (7.41)	24.13 (6.61)
words	Statistic-based	28.95 (5.71)	28.02 (5.48)
	combined approach	27.38 (7.14)	27.19 (7.07)

For evaluation, segmentation results are computed at the sentence level. We convert word (or time frame) level segmentation into sentence level one by finding the closest sentence boundary to the identified word (or time frame) boundary.

Experimental results show that our proposed algorithms have significant improvement over the base-line algorithm according to t-test at 95% confidence interval. It also can be seen that the combined approach outperforms the other algorithms.

One interesting fact that can be seen from these results is that when we use the combined approach, the results become better even than the original dynamic programming approach. This is partly because the dynamic programming approach only uses speaker distribution (Equation (10)) to compute segment score. However, more features are considered in the statistic-based algorithm, which make it capable to eliminate bad choices for boundaries, choices that have potential to mislead the original dynamic programming approach.

Additionally, from the results shown in Table II, it can be seen that there is no additional information in knowing the sentence boundaries, since the results are almost similar when the algorithms are applied to the word sequence or sentence sequence.

Finally as mentioned before, the computational cost of the original dynamic programming approach depends on the number of elements in the input sequence, $O(L^3)$. In the case that we have sentences as the input sequence, the average number of elements in the input for our test set is 610.4. When we use the combined approach and limit the number of candidate boundaries, this average is 65.2. As a result, the combined algorithm is around 90 times faster than the original dynamic programming approach when is applied on the sentence sequence. This difference is more noticeable when we compare the average number of input elements in the word sequence case. The average length of input sequence for the original dynamic programming approach is 4546.1, however this number for the combined approach is 34.2. It means that the combined approach is around 17,000 times faster than the original one. As mentioned above the original dynamic programming approach is not feasible for the word sequence case.

3) Comparison of the Algorithms (Case 2: Unknown Number of Boundaries): In a more realistic case, we assume that we have no prior knowledge about the number of boundaries. The results are shown in Table III. Like the previous case, we evaluate all the algorithms in the sentence level. We can see that our proposed approaches have significant improvement over the base-line algorithm according to t-test at 95% confidence interval. Again in this case, the best results are achieved using the combined approach. However in this case knowing the sentence boundaries and applying the algorithm on the sentence level can lead to better results than applying it on the word sequence.

4) Analyzing two cases in the statistic-based algorithm: known and unknown number of boundaries: From the results shown in Tables II and III we can see that for the statistic-based algorithm, when we have no prior knowledge about the number of boundaries, the results are better. We conducted an analysis to better understand the effect of the number of boundaries on the system performance. Note that for the two configurations (known vs. unknown number of boundaries) of the statistic-based algorithm, the candidate boundaries have the same ranking, based on their $peak\_score$ values. The only difference is where the cut off is in this ranked list when making the final decisions. The more segment boundaries the system hypothesizes, the few the missed boundaries are, and the more the false alarms are. Among the 10 meetings in our test set, we found that when the number of boundaries is unknown, the system generates more boundaries than the reference number for 4 meetings (and 3 of them have better overall p_r results), fewer boundaries for 5 meetings (1 has better overall p_r result), and the same number for 1 meeting. Overall these result in an improved p_r and p_k score.

An example (meeting *is1001c*) is shown in Fig. 10.(a). In this example, there are more segmentation boundaries predicted when the number of boundaries is unknown. The reference segmentation for this sample meeting is shown in the lower plot. In the top plot, the outputs of the statistic-based approach for two cases (known vs. unknown number of boundaries) are compared. Solid lines show the common boundaries which are found in both cases. Dashed lines show the additional boundaries which are found when the number of boundaries is not known. We can see that these additional boundaries (which have $peak\_score$ value above the defined threshold) are near true boundaries, so the found segmentation in the unknown case is better than the known one.

As a different case, Fig. 10(b) shows the result for another meeting (*is1006b*). For this one, the number of automatically determined boundaries is also more than the reference boundaries. However, the additional boundaries are not accurate, resulting in overall worse result than the case of known number of boundaries.

It is worth noting that for a different test set, the results might vary. The final performance is related to the candidate boundary ranking of the segmentation algorithm–whether the reference boundaries are correctly ranked high on the list, as

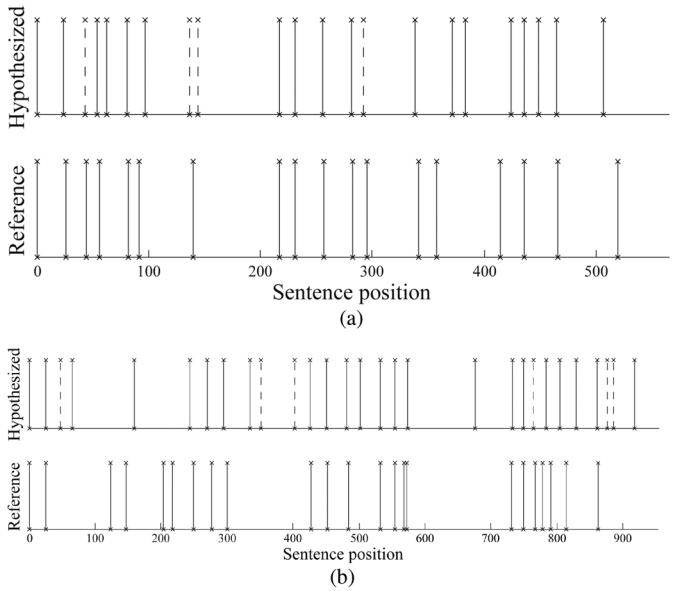


Fig. 10. Comparison of the outputs of the statistic-based approach on two sample meetings ((a) *is1001c* and (b) *is1006b*) for two cases (known vs unknown number of boundaries). The solid lines in the hypothesized plot is the common boundaries which are found in both cases. The dashed lines are additional boundaries that are returned only in the unknown case. For each meeting the reference boundaries are also shown.

well as the segmentation evaluation metrics–how the missed and false alarm boundaries are penalized in the final score.

5) Evaluating the Performance of the Candidate Boundary Detection: As described earlier, the combined method uses the statistic-based algorithm to select the candidate boundary set. It is important for this set to contain true boundaries as many as possible. In this part, we evaluate the effectiveness of the statistic-based algorithm to find this set.

First, an example of the score plot for a sample meeting (*es2008a*) is shown in Fig. 11. As can be seen from this figure, most of the true boundaries (indicated by dashed line) are located in the local maximums of the score plot. This originally motivated us to use this algorithm to find the candidate boundary set.

Second, we measure the probability for a miss of a true boundary in the statistic-based algorithm. The results are shown in Table IV. We report these results according to P_{miss} (Equation (15)) and Pr_{miss} (Equation (20)).

Finally we evaluate how much this estimation has effect on the performance of the combined algorithm. In order to do so, we add the true boundaries to the estimated candidate boundary set and feed this new set into the dynamic programming algorithm. The results are shown in Table V. For comparison purpose, we show the counterpart results from Tables II and III where the original found candidate boundary set is used. From this table it can be seen that finding a more accurate candidate set to feed to the dynamic programming approach can increase the performance of the algorithm.

6) Evaluating the Performance of Labeling Approaches: We use two different evaluation metrics in order to evaluate the performance of the introduced labeling approaches in Section II-E. First we measure the accuracy of the found category for each segment. In this sense, if the found category is different from the reference one (even according to just one participant), it is

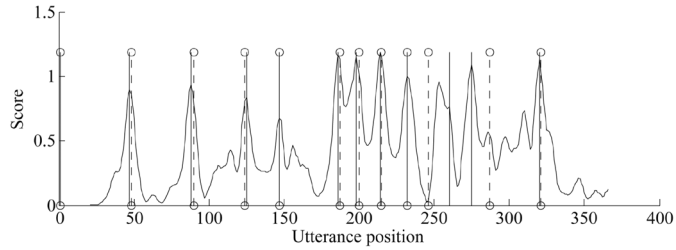


Fig. 11. The score plot for a sample meeting (*es2008a*). True boundaries are indicated with dashed line.

TABLE IV
RESULTS OF THE CANDIDATE BOUNDARY DETECTION ALGORITHM, SHOWN IN FIG. 3

Algorithm	$P_{miss}(\%)$	$Pr_{miss}(\%)$
Candidate boundary detection	8.13	12.88

TABLE V
THE PERFORMANCE OF THE COMBINED ALGORITHM WHEN THE CANDIDATE BOUNDARY SET IS THE *ORIGINAL* ONE FOUND BY THE STATISTIC-BASED ALGORITHM OR *MIXED* WITH THE TRUE BOUNDARIES

Number of boundaries	**Input type**	**Candidate set**	$P_k(\%)$	$Pr(\%)$
Known	Sentence	Mixed	23.21	21.85
		Original	26.06	23.89
	word	Mixed	16.64	16.45
		Original	25.93	24.08
Unknown	Sentence	Mixed	25.64	22.17
		Original	26.03	24.13
	word	Mixed	21.57	22.48
		Original	27.38	27.19

considered as a wrong category. Accordingly the metric Acc is defined as the percent of the correctly found categories among all the segments.

In a more relaxed criterion, we consider each participant in each segment as a different classification task. In this task, we have to identify whether the participant is active in the segment or not. Accordingly, we can define speaker-based precision (sbp), recall (sbr), and f-measure (sbf) to analyze the percent of the correctly found active participants. Specifically if a participant is correctly (mistakenly) marked active in a segment, it is considered as true positive (false positive). Similarly true/flase negatives can be defined. As an example, consider a segment with label $Discussion_{1010}$ which is labeled as $Discussion_{1100}$ by the algorithm. The first participant is truly marked as active (true positive), but the second participant is mistakenly reported as active (false positive). The third and fourth participants are examples of false and true negatives respectively. According to these definitions, we can use the conventional definition of precision, recall and f-measure to evaluate the output labels.

In order to disregard the influence of the segmentation algorithms, we apply the proposed labeling approaches on the reference segmentation and then compare the labels with the reference labels according to defined evaluation metrics. The results are summarized in Table VI.

From these results, it can be seen that while the mean-based approach finds active participants more accurate (higher precision), the ideal-based approach finds more active participants

TABLE VI
RESULTS OF THE PROPOSED LABELING ALGORITHMS

Algorithm	$Acc(\%)$	$sbp(\%)$	$sbr(\%)$	$sbf(\%)$
Mean-based	**77.44**	**94.88**	87.53	**90.84**
ideal-based	71.64	80.87	**97.62**	87.89

(higher recall). Overall, the mean-based approach outperforms the other one according to both accuracy and f-measure.

According to these results, the choice of the algorithm depends highly on the downstream application. As an example, consider a scenario where we want to use these segments in an extractive summarization task. We can eliminate the influence of all the participants other than the detected active ones in each segment. In this scenario we don't want to miss any active participants, because we will miss its influence in the corresponding segment, which can lead to an error in the extracted summary. Accordingly it is more important to have a high recall algorithm and we can use the ideal-based approach to detect the active participants.

IV. CONCLUSION AND FUTURE WORKS

In this article we presented three methods to segment a meeting into functionally coherent parts. The first one uses local information to assign a score to each possible boundary and then selects the best boundaries according to these scores. The second one uses a dynamic programming approach to find a global best segmentation according to a designed scoring function. These algorithms have their own advantages which are complementary of each other. Thus we came up with the third method which is a combination of the first two methods. We showed that these algorithms have significant improvements over our base-line method.

For our future work we aim to work more on the candidate selection algorithm, since we showed that using a candidate set which contains true boundaries can lead to a better results. This algorithm should have a good trade-off between two factors: 1) Its output candidate set must have as few elements as possible to increase the speed of the dynamic programming approach. 2) It should cover true boundaries as many as possible. We will also work on a better scoring mechanism for each segment (Equation (10)) which better reflects the goodness of a segment to be in a function segmentation. Another trend of our future work will concentrate on using these algorithms in other meeting related tasks such as information retrieval and summarization.

ACKNOWLEDGMENT

Any opinions expressed in this material are those of the authors and do not necessarily reflect the views of NSF. We thank Benoit Favre and Korbinian Riedhammer for sharing the ASR output of the AMI corpus.

REFERENCES

[1] M. Purver, T. L. Griffiths, K. P. Körding, and J. B. Tenenbaum, "Unsupervised topic modelling for multi-party spoken discourse," in *Proc. 21st Int. Conf. Comput. Linguist. and 44th Annu. Meeting Assoc. for Comput. Linguist.*, 2006, pp. 17–24.

[2] M. Purver, J. Dowding, J. Niekrasz, P. Ehlen, S. Noorbaloochi, and S. Peters, "Detecting and summarizing action items in multi-party dialogue," in *Proc. SIGdial*, 2007, pp. 200–211.

[3] W. Morgan, P.-C. Chang, S. Gupta, and J. M. Brenier, "Automatically detecting action items in audio meeting recordings," in *Proc. SIGdial*, 2009, pp. 96–103.

[4] R. Fernández, M. Frampton, P. Ehlen, M. Purver, and S. Peters, "Modelling and detecting decisions in multi-party dialogue," in *Proc. 9th SIGdial Workshop Discourse Dialogue*, 2008, pp. 156–163.

[5] L. Wang and C. Cardie, "Focused meeting summarization via unsupervised relation extraction," in *Proc. SIGdial*, 2012, pp. 304–313.

[6] D. Hillard, M. Ostendorf, and E. Shriberg, "Detection of agreement vs. disagreement in meetings: Training with unlabeled data," in *Proc. Conf. North Amer. Chapt. Assoc. Comput. Linguist. Human Lang. Technol.: Companion Vol. Proc. HLT-NAACL 2003–Short Papers-Vol. 2*, 2003, pp. 34–36.

[7] S. Somasundaran, J. Ruppenhofer, and J. Wiebe, "Detecting arguing and sentiment in meetings," in *Proc. 8th SIGdial Workshop Discourse Dialogue*, 2007, pp. 26–34.

[8] S. Raaijmakers, K. Truong, and T. Wilson, "Multimodal subjectivity analysis of multiparty conversation," in *Proc. EMNLP*, 2008, pp. 466–474.

[9] S. Germesin and T. Wilson, "Agreement detection in multiparty conversation," in *Proc. ICMI*, 2009, pp. 7–14.

[10] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing, "Discourse segmentation of multi-party conversation," in *Proc. ACL*, 2003, pp. 562–569.

[11] I. McCowan, S. Bengio, D. Gatica-Perez, G. Lathoud, F. Monay, D. Moore, P. Wellner, and H. Bourlard, "Modeling human interaction in meetings," in *Proc. ICASSP*, 2003, vol. 4, pp. 748–751.

[12] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan, "Modeling individual and group actions in meetings with layered HMMs," *IEEE Trans Multimedia*, vol. 8, no. 3, pp. 509–520, Jun. 2006.

[13] A. Dielmann and S. Renals, "Automatic meeting segmentation using dynamic Bayesian networks," *IEEE Trans. Multimedia*, vol. 9, no. 1, pp. 25–36, Jan. 2007.

[14] S. Reiter, B. Schuller, and G. Rigoll, "Hidden conditional random fields for meeting segmentation," in *Proc. ICME*, 2007, pp. 639–642.

[15] S. Banerjee and A. I. Rudnicky, "Segmenting meetings into agenda items by extracting implicit supervision from human note-taking," in *Proc. IUI*, 2007, pp. 151–159.

[16] V.-A. Nguyen, J. Boyd-Graber, and P. Resnik, "Sits: A hierarchical nonparametric model using speaker identity for topic segmentation in multiparty conversations," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguist.: Long Papers-Vol. 1*, 2012, pp. 78–87.

[17] S. Rayhan Joty, G. Carenini, and R. T. Ng, "Topic segmentation and labeling in asynchronous conversations," *arXiv preprint arXiv:1402.0586*, 2014.

[18] M. Hadi bokaei, H. Sameti, and Y. Liu, "Extractive summarization of multi-party meetings through discourse segmentation," *Nat. Lang. Eng.*, pp. 1–32, 3, 2015, FirstView.

[19] B. Webber, M. Egg, and V. Kordoni, "Discourse structure and language technology," *JNLE*, vol. 18, no. 4, pp. 437–490, 2012.

[20] D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, and G. Lathoud, "Modeling individual and group actions in meetings: A two-layer hmm framework," in *Proc. CVPR*, 2004, pp. 117–117.

[21] D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, and G. Lathoud, "Multimodal group action clustering in meetings," in *Proc. ACM 2nd Int. Workshop Video Surveill. Sens. Netw. ACM*, 2004, pp. 54–62.

[22] I. McCowan, D. Gatica-Perez, S. Bengio, G. Lathoud, M. Barnard, and D. Zhang, "Automatic analysis of multimodal group actions in meetings," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 305–317, Mar. 2005.

[23] D. Zhang, D. Gatica-Perez, and S. Bengio, "Semi-supervised meeting event recognition with adapted HMMs," in *Proc. IEEE Int. Conf. Multimedia and Expo, ICME'05*, 2005, p. 4.

[24] A. Morris, A. Hagen, H. Glotin, and H. Bourlard, "Multi-stream adaptive evidence combination for noise robust ASR," *Speech Commun.*, vol. 34, no. 1, pp. 25–40, 2001.

[25] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden Markov models for complex action recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn.*, 1997, pp. 994–999.

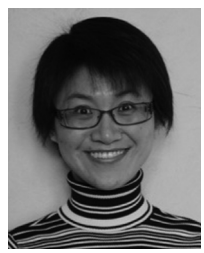
[26] S. Bengio, "An asynchronous hidden Markov model for audio-visual speech recognition," *Adv. Neural Inf. Process. Syst.*, pp. 1213–1220, 2002.

[27] A. Dielmann and S. Renals, "Dynamic Bayesian networks for meeting structuring," in *Proc. ICASSP*, 2004, vol. 5, pp. 626–629.

[28] A. Dielmann and S. Renals, "Multistream dynamic Bayesian network for meeting segmentation," *Mach. Learn. Multimodal Interact.*, pp. 76–86, 2005.
[29] S. Reiter, S. Schreiber, and G. Rigoll, "Multimodal meeting analysis by segmentation and classification of meeting events based on a higher level semantic approach," in *Proc. ICASSP*, 2005, pp. 161–164.
[30] S. Reiter, B. Schuller, and G. Rigoll, "A combined LSTM-RNN-HMM-approach for meeting event segmentation and recognition," in *Proc. ICASSP*, 2006, vol. 2, pp. 393–396.
[31] A. Tritschler and R. A. Gopinath, "Improved speaker segmentation and segments clustering using the Bayesian information criterion," in *Proc. Eurospeech*, 1999, vol. 99, pp. 679–682.
[32] M. Al-Hames, A. Dielmann, D. Gatica-Perez, S. Reiter, S. Renals, G. Rigoll, and D. Zhang, "Multimodal integration for meeting group action segmentation and recognition," in *Proc. MLMI*, 2006, pp. 52–63.
[33] 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 *Proc. ICSLP*, 2004.
[34] P. Dai, H. Di, L. Dong, L. Tao, and G. Xu, "Group interaction analysis in dynamic context," *IEEE Trans. Syst., Man, Cybern. B Cybern.*, vol. 38, no. 1, pp. 34–42, Feb. 2008.
[35] O. Brdiczka, J. Maisonnasse, P. Reignier, and J. L. Crowley, "Detecting small group activities from multimodal observations," *Appl. Intell.*, vol. 30, no. 1, pp. 47–57, 2007.
[36] S. E. Tranter and D. A. Reynolds, "An overview of automatic speaker diarization systems," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 5, pp. 1557–1565, Sep. 2006.
[37] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, and M. Kronenthal *et al.*, "The AMI meeting corpus: A pre-announcement," in *Proc. MLMI*, 2006, pp. 28–39.
[38] M. A. Hearst, "Texttiling: Segmenting text into multi-paragraph subtopic passages," *Comput. Lingist.*, vol. 23, no. 1, pp. 33–64, 1997.
[39] G. Murray and S. Renals, "Term-weighting for summarization of multi-party spoken dialogues," in *Proc. MLMI*, 2008, pp. 156–167.
[40] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.
[41] J. Puzicha, T. Hofmann, and J. M. Buhmann, "Non-parametric similarity measures for unsupervised texture segmentation and image retrieval," in *Proc. CVPR*, 1997, pp. 267–272.
[42] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.
[43] S. Renals, T. Hain, and H. Bourlard, "Recognition and understanding of meetings the AMI and AMIDA projects," in *Proc. ASRU*, 2007, pp. 238–247.
[44] J. Kolář, Y. Liu, and E. Shriberg, "Speaker adaptation of language and prosodic models for automatic dialog act segmentation of speech," *Speech Commun.*, vol. 52, no. 3, pp. 236–245, 2010.
[45] S. Quarteroni, A. V. Ivanov, and G. Riccardi, "Simultaneous dialog act segmentation and classification from human-human spoken conversations," in *Proc. IEEE Int. Conf. . Acoust., Speech, Signal Process. (ICASSP)*, 2011, pp. 5596–5599.
[46] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation," *Mach. Learn.*, vol. 34, no. 1–3, pp. 177–210, 1999.
[47] J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic detection and tracking pilot study final report," 1998.
[48] L. Pevzner and M. A. Hearst, "A critique and improvement of an evaluation metric for text segmentation," *Comput. Linguist.*, vol. 28, no. 1, pp. 19–36, 2002.
[49] M. Georgescul, A. Clark, and S. Armstrong *et al.*, "Exploiting structural meeting-specific features for topic segmentation," in *Actes de la 14eme Conf. sur le Traitement Autom. des Lang. Nat.*, 2007, pp. 15–24.
[50] M.g Purver, "Topic segmentation," in *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. New York, NY, USA: Wiley, 2011, pp. 291–317.

Mohammad Hadi Bokaei received the B.S. degree in 2008 from the Department of Computer Science, International University of Science and Technology, Tehran, Iran, and M.S. degree in 2010 from the Department of Computer Science, Sharif University of Technology, Tehran, Iran. Since 2011, he has been a Ph.D. candidate in the Speech Processing Laboratory at Sharif Univeristy of Technology. He collaborates with Dr Yang Liu's Speech and Language Processing Lab as a Visiting Student in 2014. His current research interests include natural/spoken language processing, speech summarization, spoken dialogue systems, and machine learning.

Hossein Sameti received his Ph.D. in electrical engineering from the University of Waterloo, Canada, in 1995. He received his M.Sc. and B.Sc. in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1989 and 1986, respectively. He is now a Faculty Member at the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, where he has served as the head of Artificial Intelligence group for 4 years and is currently serving as the department chair. He is the Founder and Supervisor of Speech Processing Lab (SPL) in that department, too. His research areas are speech recognition and understanding, spoken dialogue systems, speaker identification and verification, and speech enhancement.



Yang Liu (SM'13) received the B.S. and M.S. from Tsinghua University in China in 1997 and 2000, respectively, and a Ph.D. in electrical and computer engineering from Purdue University in 2004. She was a Researcher at the International Computer Science Institute at Berkeley from 2002 to 2005. She is currently an Associate Professor at The University of Texas at Dallas. Her research interests are in the area of spoken language processing, natural language processing, and machine learning.