Natural Language Engineering **22** (1): 41–72. © Cambridge University Press 2015
doi:10.1017/S1351324914000199

Extractive summarization of multi-party meetings through discourse segmentation

MOHAMMAD HADI BOKAEI[1,2], HOSSEIN SAMETI[1] and YANG LIU[2]

[1]*Speech Processing Lab, Computer Engineering Department, Sharif University of Technology, Tehran, I.R. Iran*
e-mail: bokaei@ce.sharif.edu, sameti@sharif.edu
[2]*Human Language Technology Group, Computer Science Department, The University of Texas at Dallas, Richardson, TX, USA*
e-mail: yangl@hlt.utdallas.edu

(*Received 13 April 2014; revised 1 December 2014; accepted 1 December 2014; first published online 4 March 2015*)

Abstract

In this article we tackle the problem of multi-party conversation summarization. We investigate the role of discourse segmentation of a conversation on meeting summarization. First, an unsupervised function segmentation algorithm is proposed to segment the transcript into functionally coherent parts, such as $Monologue_i$ (which indicates a segment where speaker i is the dominant speaker, e.g., lecturing all the other participants) or $Discussion_{x_1 x_2,\ldots,x_n}$ (which indicates a segment where speakers x_1 to x_n involve in a discussion). Then the salience score for a sentence is computed by leveraging the score of the segment containing the sentence. Performance of our proposed segmentation and summarization algorithms is evaluated using the AMI meeting corpus. We show better summarization performance over other state-of-the-art algorithms according to different metrics.

1 Introduction

Text summarization has a long history and various methods have been developed to generate coherent summaries of input documents. The first paper in this field dates back to 1958 (Luhn 1958). Current works on summarization are mainly focused on extractive methods where binary sentence classification outputs a cut-and-paste summary. To generate more human-like abstractive summaries, researchers have combined extractive summarization with sentence compression (either before or after summary sentence selection). Other methods have also been explored for abstractive summarization; however, the performances of the current methods are still far from perfect. More comprehensive surveys of various text summarization approaches can be found in Mani and Maybury (1999), Nenkova and McKeown (2012)and Ramezani and Feizi-Derakhshi (2014).

Improvements of automatic speech recognition systems and an increasing amount of audio data (such as broadcast news, voice mail, telephony conversations, and

meeting recordings) have attracted plenty of research interest in the field of speech summarization. This field has additional challenges compared to text summarization tasks such as speech disfluencies, recognition errors, and identifying the boundaries of language units (Liu *et al.* 2006). One specific form of speech is meetings, or multi-party conversations. Meetings are an integral part of our daily life. Not surprisingly then, there is a growing interest in developing automatic methods that summarize meetings in a way that by reading only the summary, the reader will be informed about what happened in the meeting. Compared to other speech genres such as lectures and broadcast news, meetings are especially challenging because of (i) high speech recognition error rate; (ii) many disfluencies due to its spontaneous nature; (iii) distributed information across sentences from different speakers; (iv) low information density (Buist, Kraaij and Raaijmakers 2004); and (v) lack of coherence between utterances (this can be caused by multiple participants, or the use of other modality, such as a white board).

Previous studies on meeting summarization have investigated different methods, including supervised and unsupervised approaches. Some unsupervised approaches try to assign a salience score to each utterance and then extract the most valuable ones according to these scores (Murray *et al.* 2005; Garg *et al.* 2009; Chen and Metze 2012, 2013). Other unsupervised methods do not calculate salience scores explicitly. They try to extract utterances that cover important concepts using a global optimization framework (Gillick *et al.* 2009; Xie *et al.* 2009; Riedhammer, Favre and Hakkani-Tür 2010). Supervised approaches, on the other hand, view the task of summarization as a binary classification problem. They build a model from a human annotated training set and use that model to decide about unseen utterances (Buist *et al.* 2004; Galley 2006; Xie, Liu and Lin 2008; Liu and Liu 2010). Features used in these models include lexical, prosodic, speaker, and conversational information. There are also efforts on meeting understanding from various aspects, such as *action item detection* (Morgan *et al.* 2006; Purver *et al.* 2007), *decision summarization* (Fernández *et al.* 2008; Wang and Cardie 2012), *opinion analysis* (Hillard, Ostendorf and Shriberg 2003; Somasundaran, Ruppenhofer and Wiebe 2007; Raaijmakers, Truong and Wilson 2008; Germesin and Wilson 2009), *topic segmentation* (Galley *et al.* 2003; Purver *et al.* 2006), and *meeting event detection* (McCowan *et al.* 2003). Unlike generic meeting summarization, these methods try to extract some valuable information from the meeting according to a specific perspective.

In this paper we propose to leverage discourse segmentation for meeting summarization. While some prior studies have concentrated on the influence of topic segmentation, or other discourse information such as rhetorical or structural features, on summarization methods (Marcu 1998; Maskey and Hirschberg 2003; Fung and Ngai 2006; Murray *et al.* 2006; Zhang and Fung 2012; Oya *et al.* 2014), there is no previous work that investigates the role of other kinds of discourse segmentation, namely function segmentation, on the summarization task, which is the goal of this study. First, we propose a genetic algorithm to segment a meeting into functionally coherent parts such as discussions or monologues. These parts can be considered as various events or phases (McCowan *et al.* 2003) which construct the whole meeting. Then a scoring mechanism is used to score each utterance according to

the segmentation, and top ranked sentences are extracted to form the summary. We compare our algorithm with other baseline and state-of-the-art algorithms on the AMI meeting corpus using various evaluation metrics. Results show significant improvement of our algorithm over other ones according to most of the used evaluation metrics. The main contribution of this work includes:

- An unsupervised genetic algorithm is proposed which segments a meeting functionally, and its performance is evaluated on manually annotated meeting data.
- The effectiveness of using function segmentation on meeting summarization task is studied.

The rest of the paper is organized as follows. Section 2 describes related work in summarization and discourse segmentation. An overview of our proposed idea is illustrated in Section 3. Then Section 4 describes our genetic algorithm to segment a meeting transcript, and Section 5 explains our methods for ranking the meeting utterances using defined segmentation. In Section 6, evaluation metrics and experimental results are presented. Finally in Section 7, we draw conclusions and describe our future work.

2 Related work

In this section, we review previous works for two problems that are related to our study: discourse segmentation and speech summarization.

2.1 Discourse segmentation

Language does not normally consist of isolated unrelated sentences, but rather semantically related groups of sentences. These groups are referred as discourses. Within a discourse, the whole information conveyed is more than the sum of its separate parts. In fact, patterns formed by consecutive sentences are informative too. These patterns are called *discourse structure*. Algorithms that recognize and analyze various aspects of discourse structure can be categorized into three separate layers (Webber, Egg and Kordoni 2012): *linear discourse segmentation* which aims to segment a document into a linear sequence of topically or functionally coherent parts, *discourse chunking* which refers to recognizing discourse relations between sentences such as explanation, elaboration, result, etc. (Lascarides and Asher 1993), and *discourse parsing* which resembles sentence level parsing in an attempt to construct a complete structured cover of a text (Grosz and Sidner 1986; Mann and Thompson 1988).

In this article, we focus on linear segmentation of a meeting transcription. Generally speaking, a document can be segmented from two distinct perspectives: *topics* and *functions*. Topic segmentation tries to automatically divide the document into shorter topically coherent segments.[1] The task of topic segmentation has been

[1] The definition of topic is somehow ambiguous and can be changed in different applications.

approached in many different ways. There are two basic insights that most of them use (Purver 2011):

> *Changes in content*: If we look at a discussion containing various topical segments, we might see that vocabulary being used in each segment differs dramatically from those used in other topically different segments. Many approaches are based on this idea and try to track the changes of lexical similarity between adjacent parts of the text and assign boundaries wherever this change is higher than a predefined threshold. TextTiling (Hearst 1997) and LCSeg (Galley *et al.* 2003) are two well-known methods in this category. *Distinctive boundary feature*: Boundaries themselves have their own characteristics which can be detected accordingly. For example, speakers use specific words, phrases, and even acoustic cues, called *discourse markers*, to signal topic changes to their audience. It has been shown that these cue phrases can be used to structure the given discourse (Grosz and Sidner 1986; Hirschberg and Litman 1993).

A comprehensive study of topic segmentation approaches can be found in Purver (2011).

It has been shown that LCSeg has competitive performance in the meeting segmentation task (Hsueh, Moore and Renals 2006; Purver 2011). However, it has certain weaknesses in the field of asynchronous conversations where topics are interleaved and do not change sequentially in the order of sentences. Joty, Carenini and Ng (2014) proposed extensions of this algorithm, using a feature in the surface form of asynchronous conversations, which is quoted sentences. However, this feature cannot be extracted from meeting transcripts and is not applicable in our problem.

From a distinct perspective, we can segment a document into functionally coherent parts. Differences between this kind of segmentation and topic segmentation can be well understood with an example. Suppose we have to segment a meeting transcript into shorter parts. We have two ways of doing so. The first one is to segment the transcript according to items in the meeting agenda (topic segmentation). In the second approach, we can segment the transcript into separate activities such as *discussions* between all or some of the participants, *monologues* or *presentations.* This kind of segmentation is called function segmentation (Webber *et al.* 2012).

Much of previous work on function segmentation has been performed for the genre of scientific articles, where the goal is to segment the given article into *Introduction*, *Methods*, *Results,* and *Discussion* parts (McKnight and Srinivasan 2003; Hirohata *et al.* 2008). One of the earliest work on function segmentation in the meeting domain is McCowan *et al.* (2003), which tried to recognize group actions in a given meeting transcript using an HMM-based approach. The actions to be recognized are *monologue*, *presentation*, *white board*, *discussion*, *consensus,* and *note-taking*. Similar sets of segments have been used in other studies, with segments named *group actions*, *phases* or *meeting events* (Dielmann and Renals 2004; Zhang *et al.* 2005; Reiter, Schuller and Rigoll 2007). All of the previous methods used supervised approaches in order to detect the category of each utterance. Various statistical models have also been trained and tested, such as dynamic Bayesian

network (Dielmann and Renals 2004), information fusion using multiple classifiers (support vector machines, Bayesian network, Gaussian mixture model, multilayer perceptron network, and radial basis network) (Reiter and Rigoll 2004), combination of recurrent neural network and HMM (Reiter, Schuller and Rigoll 2006), two-layer HMM (Zhang *et al.* 2004) and hidden conditional random fields (Reiter *et al.* 2007). The main problem with these approaches is that they need an annotated training set to train the underlying models. In this paper, we propose an unsupervised function segmentation algorithm.

2.2 Speech summarization

We categorize speech summarization methods into five distinct categories according to the main idea behind these methods and briefly describe them[2]. A comprehensive study of speech summarization methods can be found in Liu and Hakkani-Tür (2011).

Maximum Marginal Relevance (MMR) (Carbonell and Goldstein 1998): The main idea in this approach (and all others which are an extension of that) is to find an extractive summary consisting of utterances which are most similar to the document and least similar to the ones extracted in the summary. This algorithm is iterative and based on a query. In each iteration, an utterance is selected that is most relevant to that query and the least redundant according to utterances extracted up to that point. Relevancy and redundancy are estimated according to certain similarity measures such as cosine similarity. This algorithm is generalized to generic (query-independent) summarization (Zechner 2002a) in which the score of utterance i at iteration k is calculated as (1):

$$MMR\left(S_i^k\right) = \lambda * Sim_1(S_i, D) - (1 - \lambda) * Sim_2(S_i, Summ^{k-1}), \quad (1)$$

where D is the document vector and $Summ^{k-1}$ represents the utterances that have been selected up to iteration k. The first part of (1) measures how important the utterance is according to its resemblance to the whole document. The second part measures how redundant the sentence is based on its similarity to the extracted summary so far. Parameter λ is a tradeoff factor between importance and redundancy. Higher values for λ give more weight to more important utterances and lower values give more weight to non-redundant content.

MMR has been widely used as a competitive baseline in speech summarization (Murray *et al.* 2005). Various similarity measures and term weighting methods have been also tested and reported in the literature (Xie and Liu 2008).

Approaches based on topic modeling: The main idea here is to find the underlying topic of the meeting and then extract the utterances which cover

[2] Note that prior methods can be categorized in other ways. There are also hybrid methods which use a combination of these ideas in their approaches.

most of these topics. These approaches are based on the topic representation of documents. Several topic modeling methods exist in the literature such as latent sementic analysis (LSA) (Deerwester *et al.* 1990), probabilistic latent semantic indexing (Hofmann 1999), and latent dirichlet allocation (Blei, Ng and Jordan 2003). These methods assume that words close in meaning occur in similar context. Unlike similarity measures that only rely on word overlap, these methods represent words in a latent semantic space and thus yield a better measure of word and document similarity. Various topic modeling approaches have been evaluated for summarization, such as using LSA (Murray *et al.* 2005) and PLSI (Kong and Lee 2006).

Graph-based approaches: The idea in these methods is that the most important utterances are similar to other most important ones. In order to extract these relations, they build a graph whose nodes are utterances in the document and edges represent similarity between nodes. Utterances are scored according to the corresponding edges and scores of their neighbor utterances. Inspired by Google's PageRank algorithm (Brin and Page 1998), a random-walk procedure is applied to the constructed graph to compute the final score of each utterance (Erkan and Radev 2004; Garg *et al.* 2009; Chen and Metze 2012). Graph-based summarization methods have been shown to be very effective (Liu and Hakkani-Tür 2011).

The summarization algorithm we use in this paper (described in Section 5) belongs to this category. We also compare our approach to two other graph-based methods. One is ClusterRank (Garg *et al.* 2009). In this method, adjacent utterances are clustered according to their similarity values. A graph is then created where nodes are constructed clusters and edges are similarity values between these clusters. Then the random-walk procedure is applied to this graph to score each cluster. Afterwards, each utterance is scored according to its associated cluster score and the similarity between the utterance and the cluster. The other one constructs a two-layer graph (Chen and Metze 2012). This algorithm uses speakers' information in order to score each utterance. Utterances and speakers are represented as nodes in the utterance layer and speaker layer of the graph, respectively. Using this graph, scores from different layers are reinforced so that the final utterance scores are influenced by similar utterances as well as utterances from the same speaker.

Optimization-based approaches: The main idea in the methods in this category is to define a scoring mechanism for the extracted summaries and find the optimum point using an optimization method. These methods rely on concepts and employ optimization algorithms. They try to detect concepts in documents using simple heuristic approaches, for example, n-grams that occur more than a pre-defined threshold (Gillick *et al.* 2009). After detecting these concepts, an optimization algorithm such as integer linear programming is used to select utterances to maximize the concepts covered in the summary, under the summary length constraint (Gillick *et al.* 2009; Xie *et al.* 2009; Riedhammer *et al.* 2010).

Supervised approaches: All methods described above are unsupervised in the sense that they need no data to learn their model parameters. Summarization can be viewed as a binary classification task that aims to determine whether an utterance is important and must be included in the summary or not. Utterances can be represented by a set of features. A binary classifier is trained using labeled training data, and then used to classify test utterances as being important or not. A variety of features have been investigated for the task of summarization such as utterance length, weight, position, cue words, prosodic features, etc. (Zhang and Fung 2007; Banerjee and Rudnicky 2008; Murray and Carenini 2008; Xie *et al.* 2008). Various classifiers are experimented such as Bayesian networks, maximum entropy, hidden Markov models, conditional random fields, and support vector machines (Maskey and Hirschberg 2003; Buist *et al.* 2004; Galley 2006; Maskey and Hirschberg 2006; Zhang and Fung 2007; Xie and Liu 2010; Mehdad *et al.* 2013). The main drawback of these approaches is that they need a labeled training set. Obtaining such data is a time-consuming, expensive, and error-prone process, requiring trained human annotators and substantial amounts of supervision.

Different information sources have been used in the above summarization methods, ranging from simple surface words and sentence length to speaker and prosodic information. One important information source is discourse structure. While simple discourse features, such as utterance position or existence of specific cue words, are extensively used in the literature (Maskey and Hirschberg 2003, 2005; Murray *et al.* 2006), more complicated aspects of this kind of information are more or less neglected in previous work.

As discussed in Section 2.1, there are various layers of discourse analysis methods. The first layer is linear segmentation of the document into functionally or topically coherent parts. As stated above, topic information has been extensively used in previous work to calculate salience scores of the utterances. Explicit topic segmentation of the input document and its influence on summarization have also been studied in the literature. Oya *et al.* (2014) used a topic segmentation algorithm to detect various topics of the input meeting and then created a summary in a manner that covers all these topics. However, function segmentation and its influence on the summarization task have not been studied yet, which is the focus of this article.

The second layer of discourse analysis is extracting the relation between sentences in the document. One such information is rhetorical structure (Mann and Thompson 1988), which can be regarded as the story flow or sequence that might exist between the sentences of a document. It has been shown that effective extracting and modeling of this structure of the input document can improve the summarization task in both text (Marcu 1998; Fung and Ngai 2006) and spoken documents (Zhang, Chan and Fung 2007; Zhang and Fung 2012). Our study is similar to these that try to leverage discourse information for summarization.

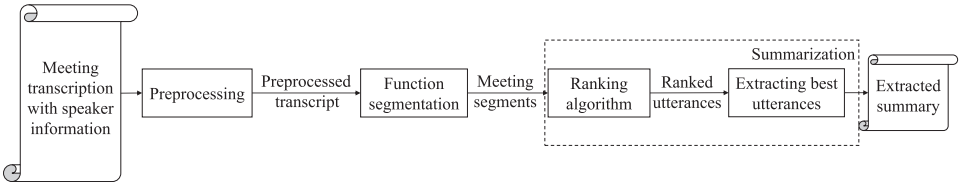


Fig. 1. The overall framework of our proposed algorithm.

3 Overview of our proposed algorithm

The main idea of our research is to exploit structural property of conversations to improve extractive summarization performance. For written text, one important structural property is the use of multiple function or semantic segments (Salton *et al.* 1997). Each segment serves an important purpose in conveying the entire document content to the reader. Detecting these segments and summarizing the document accordingly can improve summarization performance (Boguraev and Neff 2000; Neto *et al.* 2000; Angheluta, De Busser and Moens 2002). Motivated by this, the goal of this work is to study the effect of discourse segmentation on summarizing multi-party conversations.

The overall schema of our proposed algorithm is illustrated in Figure 1. In the preprocessing step, stop-words are removed and Porter algorithm (Porter 1980) is applied to stem the remaining words[3]. A segmentation algorithm is then used to segment the preprocessed transcript into functionally coherent parts. Using this segmentation as an additional knowledge source, a summarization algorithm is applied to extract the most important utterances from the transcript to form a summary.

The focus of this study is on the effect of function discourse segmentation on summarization. We hypothesize that topic-based segmentation is not as effective as function-based for the meeting summarization task. We make this hypothesis according to the fact that topic segmentation algorithms segment a transcript with respect to the changes in vocabulary used in each part. However, we noticed that these changes are very imperceptible in the multi-party conversation domain. In order to show this phenomena, we use the technique of dot-plotting which was originally used in Reynar (1994) to segment text: The discourse is plotted as a two-dimensional matrix with its words/sentences along both axes. If a word is repeated in the ith and jth position, four dots are put in (i, i), (i, j), (j, i), and (j, j) positions in the diagram. A typical example is shown in Figure 2(a) (Reynar 1994). In this figure, squares can be seen in areas with more frequent matching between near-neighbors, which correspond to topic segments. The boundaries between these squares are topic boundaries. This property is less observable in the domain of speech data. Figure 2(b) shows this diagram for a lecture transcript from a physics class (Malioutov and Barzilay 2006). The true topic boundaries are denoted by vertical lines. As this figure shows, this data is less cleanly separable, with smoother transitions between topics. However, we can still see some pale squares around the

[3] http://tartarus.org/ martin/PorterStemmer/

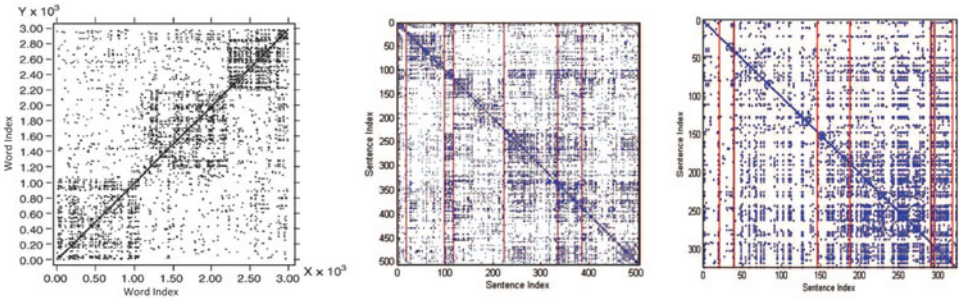


Fig. 2. (Colour online) Dot-plot diagram for (a) four concatenated Wall Street journal articles (Reynar 1994), (b) a Physics lecture (Malioutov and Barzilay 2006), (c) a meeting from AMI corpus (*es2004a*). Vertical lines indicate true segment boundaries.

diameter of the diagram. The situation is much worse in the domain of multi-party conversations. Figure 2(c) shows the dot-plot diagram for a sample meeting from the AMI corpus. In this figure there is no square at all. It shows that vocabulary usage in spontaneous conversations does not change so much. This property in the conversation domain can mislead the topic segmentation algorithms, and thus other discourse segmentation, e.g., function segmentation, may be more appropriate for the meeting domain. We will compare our function segmentation with topic-based segmentation for their impact on meeting summarization.

4 Function segmentation of multi-party conversations

We propose a genetic algorithm which aims to segment a meeting transcript into shorter parts, each one representing an event in the meeting. Genetic algorithms have been used previously to segment a written text into topically coherent parts (Lamprier *et al.* 2007) in which two criteria are used to evaluate each generated individual: *Internal cohesion* which measures the relatedness of sentences within a segment and *External dissimilarity* which measures the difference between adjacent segments. In this work, we use different criteria to segment a meeting functionally. The idea here is to separate monologue parts and various kinds of discussion ones from each other. We discriminate between parts where different individuals are involved in the meeting. Our proposed function segmentation algorithm differs from the previous methods (reviewed in Section 2.1) in two aspects:

(1) Our proposed approach is completely unsupervised, i.e., it needs no training data.
(2) Our target segment categories differ from those defined in previous works.

In this work, we define a new set of categories to segment a meeting: $Monologue_i$ and $Discussion_{x_1,\ldots,x_n}$, where i can be any one of participants in the meeting and x_i is a binary value which denotes whether speaker i involves in the discussion or not. For example, $Discussion_{1010}$ denotes a segment where speaker 1 and speaker 3 have a discussion. Note that the number of categories changes depending on the number of participants. For example, in a meeting with 4 participants we have 15 categories

(4 *Monologue*s and 11 *Discussion*s). Generally for a meeting with n participants, we have $2^n - 1$ potentially different categories. Note that for a typical meeting with more than four participants, the number of possible categories in our approach is more than that in previous works, which used ten categories regardless of the number of participants (McCowan *et al.* 2003). We define these categories for two reasons. First, since we only use meeting transcripts along with speaker information, we do not have access to information such as *presentation* and *white-board* (which are used in previous works). Second, our main goal is to exploit segmentation information for summarization. We expect that discriminating between monologue and discussion segments would help summarization. It could also be beneficial for the reader of the summary if he/she knows that the extracted utterance in the summary output is from the monologue part where a speaker lectures all the others, or from a discussion one.

In the following, we describe our function segmentation algorithm in details. First we assume that the number of segments within the input sequence is known in advance and solve the problem accordingly. Then we propose a heuristic approach which roughly estimates the number of segments in the given meeting.

4.1 General overview of genetic algorithms

Genetic algorithms can be considered as searching through the space of possible solutions to a given problem. This search process starts with an initial population considered as the first *generation*. *Individuals* in each generation are different solutions of the problem. The first step to define a genetic algorithm for a problem is to define the way in which the solutions are represented in the algorithm and converted to individuals. These individuals are then scored according to a function called *fitness function*. This function aims to evaluate the suitability of each individual to be the final solution of the problem. The best individuals are qualified to enter the next generation according to a specified *selection operator*. However, in order to search other parts of the space, qualified individuals are altered before entering the next generation according to certain *reproduction operators*. This whole process is iterated until a *stopping condition* is met. The best individual in the final generation is returned by the algorithm as the final solution of the problem. A comprehensive study of these algorithms can be found in Eiben and Smith (2003). In the following subsections, we explain these components for the function segmentation problem in more details.

4.2 Problem representation

The input to our algorithm is a sequence of speaker ids of utterances. For example, sequence *111213*... indicates that the first three utterances are uttered by speaker 1, the fourth one is uttered by speaker 2, and so on. The output of the algorithm is a segmentation of the meeting into parts in a way that each part's class differs from the previous and the next one according to our defined categories. We represent each segmentation result as a binary vector whose length is identical to the input sequence.

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

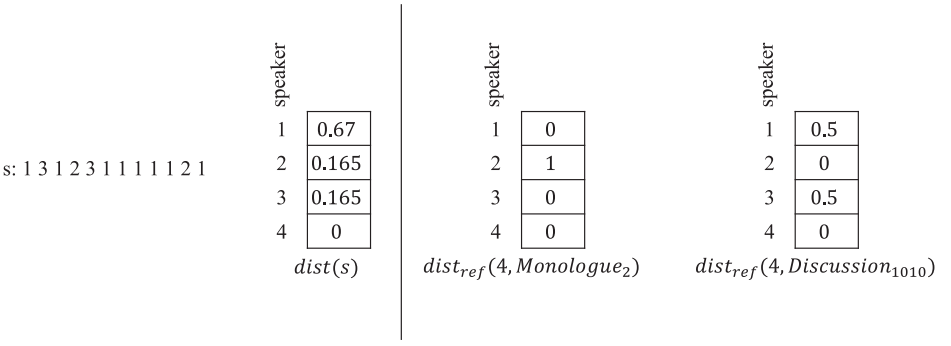


Fig. 4. Example of *dist*, $dist_{ref}$ for a sample segment.

In this vector, 1 indicates that the corresponding utterance starts a new segment. All other values are 0. Individuals in each generation in the genetic algorithm indicate various segmentations of the given sequence. These individuals are evolved according to defined operators. An example is shown in Figure 3.

4.3 Fitness function

The most important part of a genetic algorithm is its fitness function. This function must be defined in a way to have a good correlation with the ultimate evaluation criterion. We first introduce some definitions used in our algorithm. For each segment s, we define its speakers' distribution $dist(s)$ which is a vector with elements denoting the fraction of utterances spoken by each speaker in that segment. We also define a unique distribution for each category, in which all the corresponding participants are equally involved (having the same number of utterances). This is referred to as the reference distribution of category *cat* in a meeting with M participants, $dist_{ref}(M, cat)$. Figure 4 shows a sample sequence s, its distribution ($dist(s)$) and $dist_{ref}$ for two sample categories, namely $Monologue_2$ and $Discussion_{1010}$.

The idea behind our proposed algorithm is to create a segmentation consisting of segments whose distributions are more similar to the reference distributions. As an example, the sequence shown in Figure 3(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 reference distribution of category $Discussion_{011}$ ($dist_{ref}(3, Discussion_{011})$). s_2 and s_3 are similar to $dist_{ref}(3, Discussion_{101})$ and $dist_{ref}(3, Monologue_2)$, respectively.

The fitness value for each individual S with n_s number of segments is computed according to (2).

$$Fitness(S) = \frac{\sum_{i=1}^{n_s} score_{seg}(i) * length_{seg}(i)}{length(S)} \tag{2}$$

where $score_{seg}(i)$ and $length_{seg}(i)$ denote the computed score and number of utterances for the ith segment within S, respectively. This value is normalized according to the number of elements in the given sequence, $length(S)$[4] . $Fitness(S)$ can be considered as an average score for each element in the speakers sequence according to a specific segmentation S.

Now the problem is reduced to design a scoring mechanism for each segment. In order to calculate the score for a segment, we divide the segment into smaller parts each with length k. In this work, we set k to be twice the number of participants in the meeting. The score for a sample segment s is:

$$score_{seg}(s) = \frac{1}{L-k} \min_{cat \in C} \sum_{i=1}^{L-k} \{KL(dist(s(i : i+k)), dist_{ref}(N_s, cat))\}, \tag{3}$$

where L is the number of elements in the input segment s and N_s is the number of participants in the meeting. $s(i : i+k)$ is the subsegment of s with length k starting from index i. C is the list of categories defined at the beginning of this section ($Monologue_i$ and $Discussion_{x_1,\ldots,x_n}$). $KL(P, Q)$ is the Kullback–Leibler distance metric (Kullback and Leibler 1951) between two distributions:

$$KL(P, Q) = \sum_i \ln\left(\frac{P(i)}{Q(i)}\right) P(i). \tag{4}$$

Note that better segmentations result in lower fitness values. The reason behind dividing a segment into subsegments in (3) is to avoid having large segments that as a whole are similar to a reference distribution, but sub-regions have different distributions (e.g., sequence $111\ldots1222\ldots2333\ldots3$ as one segment versus three separate segments). The effectiveness of this fitness function is studied later in Section 6 and its correlation with the final evaluation metric is shown in Figure 6.

4.4 Exploration operators

Generally speaking, there are three kinds of operators which are used in a genetic algorithm: *Parent selection*, *Crossover,* and *Mutation*. Here we describe our chosen operators briefly.

4.4.1 Parent selection

We use the proportional algorithm proposed by Holland (1975) which biases the selection towards the fittest individuals. Since our problem is a minimization one, we use (5) to compute the probability of selecting individual x in generation i.

$$P_i(x) = \frac{\sum_{y \in Gen_i} 1 + Fitness(y) - Fitness^i_{min}}{1 + Fitness(x) - Fitness^i_{min}} \tag{5}$$

where $Fitness^i_{min}$ is the minimum observed fitness up to generation i (Gen_i).

[4] Note that $length(S)$ is equal for all the segmentations and does not affect the search process.

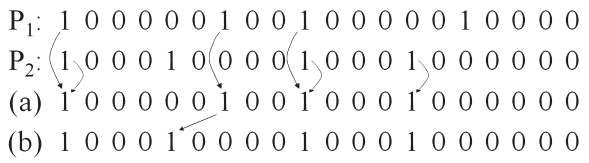


Fig. 5. Examples of applying crossover (a) and mutation (b) operators on two selected parents P_1 and P_2.

4.4.2 Crossover

When selecting two individuals (candidate segmentations) from the current generation, the crossover operator is applied to produce an offspring for the next generation. As said before, here we assume that we know the number of segments Num_{Seg}. So we have exactly Num_{Seg} boundaries (denoted by 1) in each individual. We limit the offspring to have the same number of boundaries. We select these boundaries in two steps.

(1) Boundaries that are common in two parents are transferred to offspring.
(2) The remaining needed boundaries are selected randomly from the two parents.

Mutation: We randomly choose one boundary and change its position in the offspring.

An example of applying these operators is depicted in Figure 5. In this figure, we assume that each individual has exactly four boundaries. The crossover operator's result is shown in Figure 5(a). This operator first transfers the common boundaries in two parents (position 1, 10) and then chooses the remaining needed boundaries randomly (position 7, 14 which are selected from the first and second parent, respectively). The mutation operator is shown in Figure 5(b). This operator randomly changes the position of an arbitrary boundary (boundary at position 7 is transferred to position 5).

4.5 Stopping condition

There are various stopping conditions proposed in the literature (Engelbrecht 2007). We monitor the best individual in each generation and terminate the algorithm when no improvement is observed over a number of consecutive generations.

4.6 Additional features

We use two optional features in our genetic algorithm in order to improve its effectiveness:

(1) We use *Elitism* which refers to the process of ensuring that the best individuals of the current population survive to the next generation. We select R_{eli} percent of the population that achieve best fitnesses and copy them to the new population without any changes. The remaining individuals are generated using exploration operators illustrated in Section 4.4.

(2) Premature convergence is a phenomenon that occurs when a population for an optimization problem converges too early, resulting in a suboptimal answer. In order to prevent this problem, we use *Fitness sharing* that is originally introduced as a population diversity maintenance technique and increases the exploration capability of the algorithm (Goldberg and Richardson 1987). In order to measure distances between individuals, we use *Edit distance* formulated by Levenshtein (1966), which is a popular method for measuring the difference between two strings. In this framework, we consider segmentation vectors as binary strings and use (6) to compute the final fitness value for each individual x in the ith generation Gen_i.

$$f_i(x) = \frac{|Gen_i| * Fitness(x)}{\sum_{y \in Gen_i} Edit\_Distance(x, y)} \tag{6}$$

$Fitness(x)$ is calculated according to (2) and $|Gen_i|$ denotes the population size.

4.7 Complete algorithm

The complete algorithm is shown in Algorithm 1. In this algorithm, $U(0,1)$ is a function which generates random numbers ranging between 0 and 1. ρ_c and ρ_m are crossover and mutation rate, respectively. Typically very low values for ρ_m and relatively high values for ρ_c are recommended. We set ρ_m and ρ_c to be 0.1 and 0.9, respectively. All the other parts of the algorithm were explained before.

4.8 Estimating of the number of segments

So far we have assumed that the number of segments is known and solve the segmentation problem of segmentation accordingly. However this is a very restrictive assumption which is not true in reality. Here we propose a heuristic method in order to estimate the number of segments in a given sequence. We use speakers' distributions defined in Section 4.3 in a greedy framework and try to detect various parts of the sequence where these distributions change. In this approach, we generate a base segmentation and merge consecutive segments until the Kullback–Leibler distance (Kullback and Leibler 1951) between the merged segment and the original one exceeds a predefined threshold. The complete algorithm is shown in Algorithm 2. This algorithm has two parameters. Parameter T is used to generate the initial segmentation in which the input sequence is segmented into equal T length parts. TH_{cu} is the threshold we use to merge consecutive segments. In this work, we consider T and TH_{cu} to be 5 and 0.22, respectively. We use the number of segments in the output segmentation in order to use the genetic algorithm to segment the sequence functionally.

The output segmentation from this heuristic algorithm is used by our genetic algorithm in two ways: the number of segments in this segmentation is used as the estimated number of segments (Num_{seg} in Algorithm 1). The output itself is included in the initial population of our genetic algorithm. This heuristic method together with random segmentation act as the baselines to which we compare our genetic function segmentation results.

Algorithm 1 Genetic algorithm for function segmentation of a meeting with known number of segments

Input: *seq* (Sequence of speaker id's of a meeting), Num_{seg} (Number of segments)
Parameters: N (Population size), R_{eli} (Elitism ratio), i (Generation counter), ρ_c (Crossover rate), ρ_m (Mutation rate).
$i = 0$
Create initial population $C(i)$ by generating N random segmentations each having Num_{seg} number of segments.
while stopping condition not met (Section 4.5) **do**
  Evaluate fitnesses of individuals in $C(i)$ using (6)
  $C(i+1) \leftarrow N * R_{eli}$ of the best individuals in $C(i)$
  for $j = 1 : N - N * R_{eli}$ **do**
    $[P_1, P_2] \leftarrow$ Select parents from $C(i)$ according to Section 4.4
    $ind_c \leftarrow best(P_1, P_2)$
    if $U(0,1) < \rho_c$ **then**
      $ind_c \leftarrow Crossover(P_1, P_2)$ according to Section 4.4
    end if
    $ind_m \leftarrow ind_c$
    if $U(0,1) < \rho_m$ **then**
      $ind_m \leftarrow Mutataion(ind_c)$ according to Section 4.4
    end if
    $C(i+1) \leftarrow ind_m$
  end for
  $i = i + 1$
end while
Output: best individual in the last generation

Algorithm 2 Heuristic function segmentation

Input: *seq* (Sequence of speaker id's of a meeting)
Parameters: T, TH_{cu}
seg $\leftarrow$ Segmentation of *seq* into T length parts.
while No changes occurred in *seg* **do**
  $num_{seg} \leftarrow$ Number of segments in *seg*
  for all segment s in *seg* **do**
    $dist_s \leftarrow$ speakers distribution for s
    Merge s with its consecutive segments until distance between the merged segment and s reach TH_{cu}
  end for
end while
Output: Segmentation *seg*

5 Summarization algorithm

Our summarization algorithm consists of three steps. First, we compute the salience score for each segment using the graph-based framework. Next, the importance of

each utterance is evaluated according to the score of the corresponding segment and the similarity between the utterance and the segment. Finally, summary sentences are selected. In the following, these steps are explained.

5.1 Segment score

In the first step, graph G is constructed in which each segment is represented as a node and directed edges between nodes are weighted according to their similarity. We treat each segment as a bag of words and use the normalized cosine similarity measure:

$$weight_{edge}(X, Y) = \frac{cosine(X, Y)}{\sum_Z cosine(X, Z)} \tag{7}$$

where $cosine(X, Y)$ is computed according to (8).

$$cosine(X, Y) = \frac{\sum_{w \in X,Y} tfidf(X, w) * tfidf(Y, w)}{\sqrt{\sum_{w_x \in X} tfidf(X, w_x)^2} \sqrt{\sum_{w_y \in Y} tfidf(Y, w_y)^2}} \tag{8}$$

$tfidf(X, w)$ is the term weight for word w in segment X. It is the product of term frequency of w in X and the inverse document frequency (inverse of the number of all segments containing w). We apply the random-walk process on G to compute the salience score for each segment. Specifically, (9) is used iteratively to compute each segment's score.

$$P(u) = \frac{d}{N} + (1 - d) \sum_{v \in adj(u)} weight_{edge}(u, v) * P(v), \tag{9}$$

where P is a vector whose elements are the salience scores for the segments. d is damping factor that is typically chosen in the interval $[0.1, 0.2]$ and ensures convergence (Erkan and Radev 2004). u and v are two nodes of G and $adj(u)$ denotes the nodes that are adjacent to u in graph G. Vector P can be chosen randomly at first and (9) is then iteratively applied until a stopping criterion is met (e.g., change of vector P is less than a predefined threshold ε).

5.2 Sentence scoring

Next, we compute the utterance scores by leveraging the segment scores. Similar to previous work (Garg *et al.* 2009), the score for an utterance U is its importance score in the associated segment S (computed using the cosine similarity between U and S), weighted by the segment score, that is,

$$score(U) = cosine(U, S) * P(S), \tag{10}$$

where P is the segment score vector, computed according to (9).

5.3 Summary sentence selection

After we obtain the salience score for each utterance, the last step is to select summary sentences. One simple approach would be just selecting the top ranked n sentences.

However, redundancy may exist among them. Because of the spontaneous nature of the uttered sentences in a meeting, there may be many utterances resembling to each other. For example, this can occur when a speaker wants to emphasize a previously uttered sentence, or asks a question about a certain previous subject using the same wording. Our algorithm, like all other summarization algorithms, mainly depends on the words used in the utterance. Therefore, two utterances with the same wording get similar scores using the scoring algorithm described earlier. To avoid selecting similar sentences, we use a re-ranking algorithm to decrease the score of an utterance according to its similarity to its above ranked ones. Suppose that U_i is the ith ranked utterance according to the above scoring algorithm. The new score for this utterance is calculated with respect to all the above utterances in the ranked list according to (11).

$$new\_score(U_i) = score(U_i) - \lambda * \sum_{j=1}^{i-1} \left(\frac{cosine(U_i, U_j)}{Length(U_j)} \right). \tag{11}$$

In this equation, $score(U_i)$ is the previous calculated score as illustrated in (10), $cosine(U_i, U_j)$ is the cosine similarity between two utterances (8), $Length(U_j)$ is the number of words in the utterance j, and λ is a parameter which controls the degree of redundancy in the output summary. We set λ to be 0.1 in this work. Using these new scores, the chance of selecting similar utterances is reduced. For the final summary, we greedily select the top ranked sentences.

6 Experimental results

In this section, results of applying the proposed algorithms are reported. First, the effectiveness of our function segmentation algorithm is evaluated on a manually annotated meeting test set. Then our entire summarization system is compared with other popular and state-of-the-art summarization methods.

6.1 AMI meeting corpus

The AMI meeting corpus (Carletta *et al.* 2006) is a multi-modal data set consisting of 100 hours of meeting recordings. This corpus includes speech audio, transcripts, and human extractive summaries. There are both scenario-based and non-scenario (naturally occurring) meetings. In order to test the whole proposed summarization algorithm, we use the scenario portion of the corpus in which the participants play different roles to discuss various aspects of designing a remote control. While the scenario given to participants is artificial, the speech, actions and decisions are completely spontaneous and natural. There are four different roles assigned to participants: *project manager*, *Marketing expert*, *User interface designer,* and *Industrial designer*. Project manager runs the meetings, produces and distributes minutes, and produces a report at the end of the trial. These reports are then used by annotators to determine the reference extracted summary for each meeting. They mark dialog acts that support material in the project manager's reports. There are other annotations that are available in this corpus, such as topic segmentation, head

and hand gestures, location of individuals, focus of attention, etc. For each group of participants, there are four phases with different goals: *project kick-off* consisting of getting familiar with each other and the task; *Functional design* in which user requirement, the technical functionality and the working design are set; *Conceptual design* in which the conceptual specification for the component is determined; *Detailed design* which finalize the product and evaluate the result. Each phase is prepared as a separate meeting in the corpus with the same ID. There is a total of 137 meetings in this corpus where reference extracted summaries are prepared.

In order to evaluate the effectiveness of our proposed segmentation algorithm, a subset of the meetings in the AMI corpus were manually annotated and used as our test set. It contains 11 meetings with these ids: *es2008a*, *is1000a*, *is1001a*, *is1001b*, *is1001c*, *is1003b*, *is1006b*, *is1008a*, *is1008b*, *is1008c,* and *ts3005a*. 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 work. We employed graduate students as annotators and asked them to segment the meetings according to their different events. They were given a guideline which includes 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 last annotated meeting (*ts3005a*) was used as our development set on which we tuned the parameter TH_{cu} of the heuristic algorithm.

To evaluate the whole summarization algorithm, we use 20 meetings in these series: *ES2004*, *ES2014*, *IS1009*, *TS3003,* and *TS3007*. For each meeting, 3 human generated reference extracted summaries are prepared in the corpus. Theses extractive summaries have no unique compression ratio.

6.2 Function segmentation evaluation

6.2.1 Evaluation metric

There are various segmentation evaluation metrics proposed in the literature (Pevzner and Hearst 2002; Georgescul, Clark and Armstrong 2009). We use P_k (Beeferman, Berger and Lafferty 1999), which is the most widely used metric for segmentation evaluation. 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. Note that P_k is a measure of error and thus a lower score means better segmentation performance. The formula for P_k is shown in (12).

$$P_k = \frac{\sum_{i=1}^{N-k} \delta_H(i, i+k) \oplus \delta_R(i, i+k)}{N-k} \tag{12}$$

where H is the system generated segmentation and R is the reference segmentation. Given a segmentation S, $\delta_S(i, j)$ is a function which outputs 1 if and only if segmentation S assigns ith and jth element to the same segment. 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,

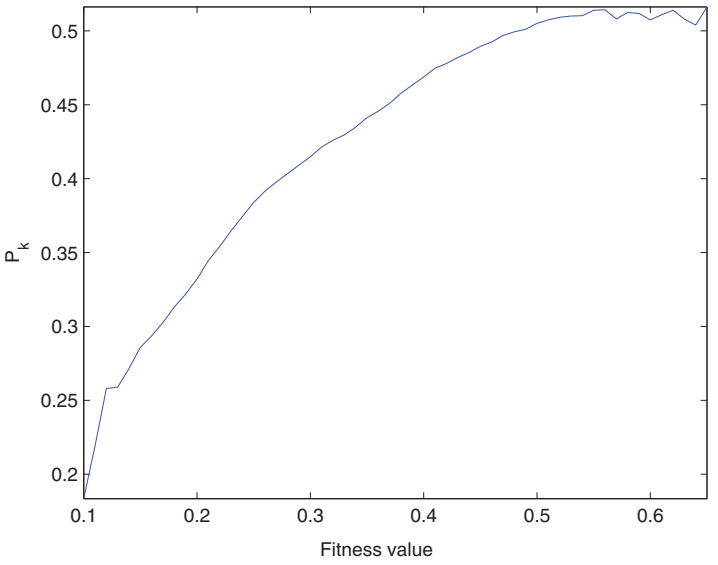


Fig. 6. (Colour online) Correlation between our designed fitness function and ultimate evaluation metric P_k.

evenly spaced boundaries, or randomly spaced boundaries) all have $P_k = 0.5$ (Purver 2011). For our evaluation process, we use the same choice for k.

6.2.2 Results

First we evaluate the effectiveness of our proposed fitness function (in (2) and (3)), by measuring its correlation with the evaluation metric P_k. We compute P_k and the fitness value for thousands of various kinds of segmentations. For each fitness value, the associated P_ks are averaged. Results are shown in Figure 6. As shown in this figure, there is a good correlation between these measures. Hence, we expect that a genetic algorithm that minimizes this fitness value will minimize the ultimate evaluation metric and thus will improve the segmentation.

Results of applying different segmentation algorithms on the test set are shown in Table 1. Since there is no prior work that is completely comparable to ours, we compare our algorithm with two baseline algorithms: random segmentation and the heuristic approach shown in Algorithm 2. We can see that our algorithm achieves the best performance. When the number of segments is automatically determined, there is a performance degradation compared to using the reference number of segments, which is expected. Table 1 shows that running the genetic algorithm improves results upon the initial condition, which is the output from the heuristic segmentation algorithm.

We also examine the effect of the approximation used in the genetic algorithm. Using the same scoring function for a segment (3), a dynamic programming algorithm can be used to find the exact solution in polynomial time. Specifically we define $score_{best}(i,k)$ to be the score of the best found segmentation for the first i elements

Table 1. *Results of our proposed genetic algorithm (GBS) compared to random segmentation and the simple heuristic method introduced in Algorithm 2*

Algorithm	P_k
Random segmentation	0.50
Heuristic segmentation (Algorithm 2)	0.43
GBS (known number of segments)	0.33
heuristic + GBS (unknown number of segments)	0.36
Dynamic programming (known number of segments)	0.31

of the given sequence using k segments. We can define the recursive formula as (13):

$$score_{best}(i,k) = \min_{j<i}\{score_{best}(j,k-1) + score_{seg}(j+1:i)\}, \tag{13}$$

where $score_{seg}(j+1:i)$ is the score of the subsequence $j+1$ to i to be considered as one segment according to (3). We also store the best j to backtrack to find the optimal path.

The initialization can be done using the (14):

$$score_{best}(i,1) = score_{seg}(1:i) \qquad \forall i \leq L, \tag{14}$$

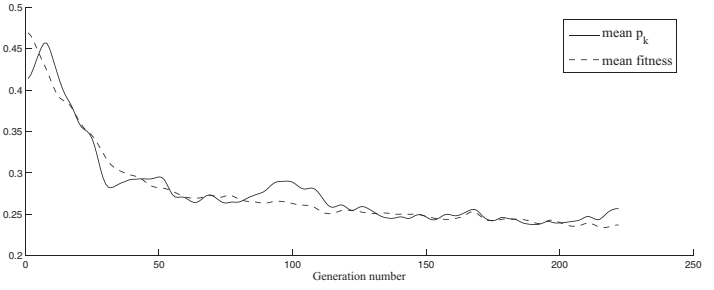
where L is the number of elements in the given sequence.

Initializing with (14) and then iteratively applying (13), we can reach to $score_{best}(L, Num_{seg})$ which is the score of the best found segmentation according to the defined score function. Experimental results show that the performance of this exact solution on the same test set is 0.31, assuming that the number of boundaries is known in advance. However, the computation cost of this approach is much higher than the genetic algorithm. If we consider the most time-consuming function in these approaches to be the calculation of $score_{seg}$ (3), in the dynamic programming approach, the number of times that this function is called is $O(Num_{seg} * L^2)$ where Num_{seg} is the number of segments and L is the length of the sequence (the average value for L in our test set is 610). However, this number in the genetic algorithm is $O(Num_{seg} * N * Num_{iter})$ where N is the population size (which is considered to be 20 in our experiments) and Num_{iter} is the number of iterations (which is typically less than 100). As a real example, for an input sequence with 600 elements on a system with core i7 CPU and 4 Gig RAM, the rough time to complete the genetic algorithm is around 30 seconds. But the dynamic programming algorithm takes around 20 minutes to complete the task.

Figure 7 shows the results of applying the genetic algorithm on a sample meeting (es2008a). Figure 7(a) shows the boundaries found by our proposed algorithm (dashed lines) and the reference boundaries (solid lines). It can be seen that the algorithm succeeds in finding most of the boundaries.

Figure 7(a) shows how the results change over generations in the genetic algorithm. In this figure, the x axis is the generation number and the y axis is the average score obtained by either the evaluation metric (P_k) or the fitness function defined in (2) for top five best individuals in each generation. From this figure, it can be seen that in general, P_k values of the best individuals decrease, which implies that better

(a)



(b)

Fig. 7. Results of applying the segmentation algorithm on the sample meeting *es2008a*: (a) Location of found boundaries (dashed line) in comparison with the reference boundaries (solid line). (b) Results (mean P_k and mean $fitness$) over generations.

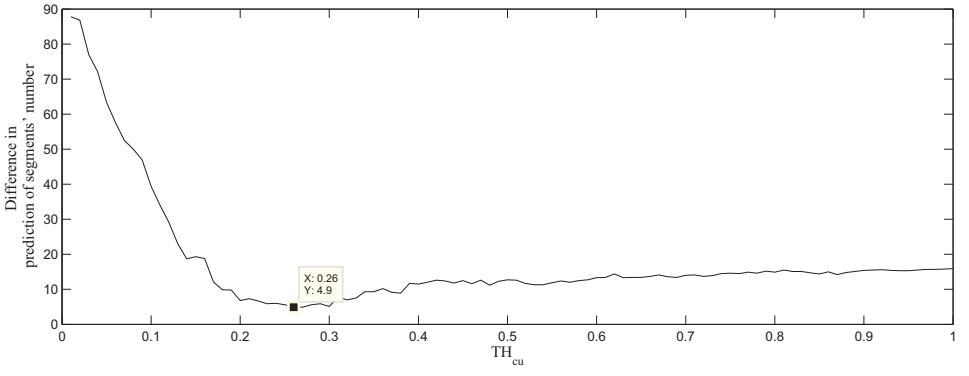


Fig. 8. (Colour online) The difference between predicted number of segments and the actual number in the reference segmentation according to various values for the heuristic algorithm's parameter TH_{cu}.

solutions are achieved in consecutive generations. After around 110 iterations, there is no more significant improvement of the segmentations found by the algorithm according to both the fitness value and P_k.

We also evaluate the performance of the heuristic algorithm in estimating the number of segments in the reference segmentation. We calculate the average difference between the hypothesized number of segments in the output of the heuristic algorithm and the actual one in the reference segmentation over the test set using different values for the parameter TH_{cu}. Figure 8 shows the result. As shown in this figure, the best prediction is achieved when $TH_{cu} = 0.26$. Note that we use $TH_{cu} = 0.22$ because we tune this parameter according to just one separate meeting.

6.2.3 Discussion

There are various advantages for using a genetic algorithm for the segmentation task:

(1) It is completely unsupervised. It needs no training data to learn its parameters. In this manner it will not depend on a specific domain and can be applied to various types of meetings.
(2) Genetic algorithms have the power of focusing on portions of the search space (which is a huge space in our problem) in which partially good segmentations have been found, and then try to improve these partially good segmentations (exploitation power of a genetic algorithm). At the same time they look at the other portions of the search space for other good candidates (exploration power of a genetic algorithm). According to this capability, we expect that the best solution found in each generation becomes better and better in consecutive generations. This can also be conveyed from Figure 7(b).
(3) Genetic approaches evaluate the segmentation of the whole sequence rather than setting boundaries incrementally.
(4) If a fitness function is a good approximation of the goal function that an optimization problem tries to maximize or minimize, genetic algorithms can be used to find (at least) suboptimal answers for the problem. In this work, we found a fitness function that well approximates the ultimate goal function (p_k measure). This can be conveyed from Figures 6 and 7(b). In these figures we see that the values of the two functions (p_k and fitness) correlate very well. Accordingly we can expect that the genetic algorithm that improves the fitness function (according to its exploitation and exploration capabilities) can also improve the ultimate goal function.

However, the major disadvantage of the genetic algorithm is its dependency on the initial generation. This dependency can be relieved by starting the algorithm with more appropriate and distributed individuals and also by increasing the exploration power of the algorithm (e.g., increase the number of individuals in each generation, apply heavier operators, etc.). Currently we run the algorithm multiple times with different initial generations and select the segmentation which is more repeated in these runs.

6.3 Summarization evaluation

6.3.1 Evaluation metric

Evaluation of summarization systems is a hard task in part due to the difficulty of creating a unique gold standard. Even human-generated summaries are different from each other. In an experiment conducted in Mani *et al.* (2002), it is shown that Kappa statistics of human agreement on generated summaries is approximately 38%. In this article, we use two classes of evaluation metrics in order to compare our results with other state-of-the-art algorithms.

According to the fact that extractive meeting summarization is indeed a classification task where important utterances must be distinguished from not important

ones, we can use common evaluation metrics in classification tasks such as precision, recall and F-measure. Since there is no unique summary for each meeting in this corpus, we use weighted version of these metrics originally introduced in Murray *et al.* (2006). For each test meeting, the system output will be compared with the reference summary set, and weighted precision, recall and F-measure at the sentence level will be computed.

The second class of evaluation metrics used in this article is *ROUGE* (Lin 2004) which evaluates a summarization system based on the number of overlapping units such as n-grams, between the system generated summary and the reference ones. This measure has been widely used in speech summarization evaluation tasks. We show results of our proposed algorithm using ROUGE-1 (unigram overlap) and ROUGE-SU4 (skip-bigram with maximum gap length of 4). It is shown that other kinds of scores in ROUGE are always highly correlated to either of them in meeting summarization (Liu and Liu 2008).

6.3.2 Results

We compare our proposed function-based summarization (FBS) with other baseline algorithms:

- *Length-Based*: It is previously shown that the utterance length is the best feature for indicating summary sentences (Murray *et al.* 2006). In this simple approach, utterances are sorted according to their length and the best ones are extracted until the compression ratio is achieved.
- *ClusterRank* (Garg *et al.* 2009): As explained in Section 5, our scoring mechanism is inspired by this algorithm.
- *MRRW-WBP* (Chen and Metze 2012): This is a state-of-the-art meeting summarization algorithm that shows an improvement over other baseline algorithms.
- *LSA* (Gong and Liu 2001), *MMR* (Carbonell and Goldstein 1998): These are popular methods that are widely used as competitive baselines in the literature.

All these algorithms (except length-based which is an obvious approach) are reviewed in Section 2. We additionally compare our proposed algorithm with two other methods based on topic segmentation. We call them topic-based summarization (TBS). The summarization method is the same as our FBS algorithm. The only difference is in the way how the segmentation is done. The first one uses TextTiling algorithm (Hearst 1997), and the other one uses LCSeg algorithm (Galley *et al.* 2003).

For each meeting in the test set, an extractive summary is generated according to multiple compression ratios (6%, 10%, 20%). This generated summary is compared with the reference set in AMI meeting corpus. Results are shown in Table 2. These results show that our proposed method is superior to other methods according to most evaluation metrics and all compression ratios. As explained in Section 5, our scoring mechanism is inspired by the ClusterRank algorithm. So we consider

Table 2. *Results of different summarization methods with Compression Ratio = 6%, 10%, and 20%. * denotes a significant improvement over the ClusterRank algorithm according to t-test at 95% confidence interval*

		Classification (%)			ROUGE-1 (%)			ROUGE-SU4 (%)		
	Algorithm	P	R	F	P	R	F	P	R	F
6%	length-based	57.82	4.87	8.97	75.03	20.16	31.44	48.49	3.65	6.68
	LSA	39.93	**6.49**	11.13	72.08	17.64	28.20	51.87	3.13	5.87
	MMR	51.69	4.67	8.54	69.04	18.79	29.19	42.69	3.41	6.20
	ClusterRank	51.83	5.57	10.03	76.56	20.47	31.95	60.29	4.51	8.24
	MRRW-WBP	60.66	5.24	9.62	74.96	20.65	31.93	48.53	3.91	7.07
	TBS(TextTiling)	58.33	5.73	10.41	77.77	20.77	32.45	60.88	4.56	8.34
	TBS(LCSeg)	56.78	5.84	10.56	77.19	20.14	31.62	60.83	4.28	7.88
	FBS	**62.14***	6.26	**11.33***	**79.52***	**21.64***	**33.70***	**64.09***	**4.94**	**9.02***
10%	length-based	55.79	8.27	14.35	72.01	30.67	42.44	44.28	8.40	13.69
	LSA	31.95	10.36	15.57	64.89	23.23	33.86	41.57	5.53	9.57
	MMR	52.86	8.36	14.36	69.13	30.19	41.41	40.90	8.18	13.17
	ClusterRank	49.44	8.79	14.87	77.45	28.86	41.66	54.10	7.70	13.22
	MRRW-WBP	58.09	9.34	16.01	73.97	31.98	44.02	46.51	9.08	14.70
	TBS(TextTiling)	56.43	9.82	16.67	74.83	31.96	44.19	57.14	10.87	17.65
	TBS(LCSeg)	54.35	9.85	16.61	73.64	31.37	43.53	56.41	10.48	17.21
	FBS	**59.23***	**10.62***	**17.90***	**76.06***	**33.18***	**45.74***	**59.15***	**11.49***	**18.75***
20%	length-based	52.06	18.57	27.24	66.69	54.92	59.42	37.79	26.23	29.46
	LSA	32.87	21.07	25.48	60.89	43.91	50.43	37.16	19.50	24.54
	MMR	50.77	19.01	27.53	65.42	55.34	59.11	35.35	26.02	28.45
	ClusterRank	36.42	**23.30**	28.26	67.71	50.76	57.40	**47.83**	26.90	33.11
	MRRW-WBP	**53.90**	20.14	29.18	**68.21**	56.37	60.89	38.78	27.01	30.32
	TBS(TextTiling)	50.39	20.56	29.04	67.45	54.93	59.75	46.71	31.03	35.48
	TBS(LCSeg)	50.46	21.61	30.09	66.99	55.07	59.80	46.08	30.92	35.52
	FBS	52.19*	22.22	**30.95***	68.18*	**56.56***	**61.13***	47.79	**32.79***	**37.24***

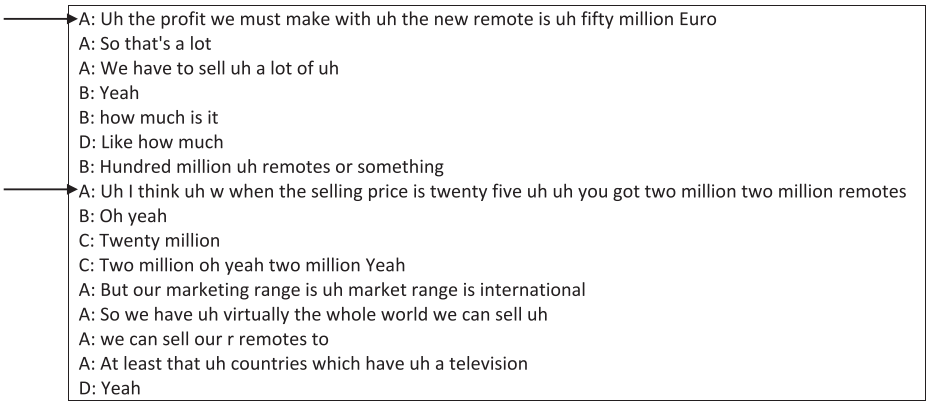


Fig. 9. A segment from meeting *TS3007a* which shows local important sentences that are detected by our proposed algorithm but not with the base-line algorithms

this algorithm as our baseline and perform statistical significance test between it and our FBS. From these results, it can be seen that our proposed method improves over ClusterRank algorithm significantly. Even compared with MRRW-WBP, which had the best performance among previous works (Chen and Metze 2012), our proposed approach outperforms it too according to most evaluation metrics. For different discourse segmentation methods, we can see that using our function segmentation yields better summarization performance than the topic-based segmentation methods.

6.3.3 Discussion

The major advantage of our proposed algorithm over other methods is its capability to detect local important utterances. The importance of these utterances is due to the context where they are used. An example is shown in Figure 9. In this figure, two utterances marked with arrows are detected only by our proposed algorithm. The similarity between these utterances and the whole document is not very much. Algorithms that calculate the score of each utterance according to its similarity to the whole document cannot detect them, instead our proposed algorithm segments the transcript and then calculates the score of each utterance according to its similarity to the corresponding segment. Because of their similarity to their associated segments, these local important utterances are detected by our algorithm to be included in the extracted summary.

In the following, we show various errors resulted by our proposed summarization approach, categorized into four different groups. Note that many of these errors are not unique to our algorithm and they are also seen in almost all the other algorithms.

(1) *Missing short summary utterances*: Many errors are caused by short utterances. These short utterances are not informative by themselves, however their relation with other utterances makes them important. The following shows some examples. Each example is marked as important by the reference

summary, but our proposed algorithm failed to detect and extract them into the summary.

- *Uh this is the agenda for today.*
- *Uh there are three stages.*
- *You can guess what it is.*

There are also utterances that are not related to other important ones, but are just extracted for the sake of coherence in the reference extracted summary. Examples are listed below:

- *Welcome at the kick-off meeting.*
- *uh and uh I think it's very uh good to introduce uh yourself.*

(2) *Incorrect inclusion of utterances due to utterance scoring based on segmentation information*: One feature of segmentation-based algorithms (like FBS and ClusterRank) is that they add a local perspective in order to decide the importance of an utterance. They consider the whole text to estimate scores for each segment (global view) and then they use a local view and estimate the importance of each utterance according to other utterances in the same segment. While this feature can be beneficial for detecting locally important utterances, it can cause another kind of errors, partly because spontaneous speech, *repetition* happens quite often – generally the wording of an utterance can influence the ones used for the upcoming utterances, which results in similar consecutive utterances. Below is such an example (the bold one is extracted by our proposed algorithm)

- *what are we going uh to uh to do.*
- ***What are we going to uh uh make f uh kind of functions in the remote.***
- *And why are we going to do it.*

This repetition will increase the score of utterances, because the similarity of these utterances to the whole segment will increase. However this is not necessarily indicative of importance. In the above example, the second utterance is extracted by our proposed algorithm, but is not marked as important in the reference summary.

(3) *Missing important utterances because of the redundancy removal step*: Another category of errors is made by the redundancy removal algorithm. Sometimes two utterances are similar to each other and are both in the reference summary. However, the redundancy removal algorithm detects them and decreases their scores. Consequently they may not be both extracted in the summary. An example is shown below:

- *Um and as you uh may have heard the documents in the shared folder uh can be uh showed on this screen Not in y the My Documents.*
- *So if you wanna show something put it in the shared folder.*

In this example, both utterances are marked as important in the reference summary. However only the first one is extracted by the system and the second one will drop on the ranked list after applying the redundancy removal algorithm.

(4) *Utterances with the same meaning but different surface form*: There are some utterances that have the same meaning, but one of them is included in the reference summary and the other one is selected by our algorithm. An example is shown below:

- *What you think might be uh a useful uh new feature.*
- *What uh what can distinguish our new trendy remote control from all the others.*

The first utterance is selected by the human annotator, and the second one is extracted by our algorithm. While they are not different in nature, this does have an impact on the system performance based on the currently used performance metrics.

7 Conclusion and future works

In this work, we investigated the effect of discourse structure for meeting summarization. We proposed a genetic approach in order to segment a meeting transcript into functionally coherent parts. The definition and goals of this kind of segmentation were illustrated and results were shown. Based on this segmentation, we proposed a complete summarization algorithm to extract important utterances from a meeting. The efficacy of our proposed summarization algorithm was compared with other state-of-the-art algorithms. Experimental results confirmed the effectiveness of our proposed algorithm.

There are a few directions in our future work. First, we plan to investigate other function segmentation algorithms. Second, we will explore other levels of discourse analysis such as extracting relations between utterances for summarization. Third, in the summarization module, we will use other term weighting methods (other than $tfidf$), as well as develop summarization algorithms that better exploit the segmentation information. Finally, we will conduct human evaluation of the summarization results and design a meeting summarization interface that allows users to obtain summaries for different segments (e.g., monologues, discussions).

References

Angheluta, R., De Busser, R., and Moens, M.-F. 2002. The use of topic segmentation for automatic summarization. In *Proceedings of the ACL-2002 Workshop on Automatic Summarization*, New Brunswick, NJ: ACL, pp. 66–70.

Banerjee, S., and Rudnicky, A. I. 2008. An extractive-summarization baseline for the automatic detection of noteworthy utterances in multi-party human-human dialog. In *Proceedings of the Spoken Language Technology Workshop*, Goa, India: IEEE, pp. 177–180.

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

Blei, D. M., Ng, A. Y., and Jordan, M. I. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research* **3**: 993–1022. MIT Press.

Boguraev, B. K., and Neff, M. S. 2000. Discourse segmentation in aid of document summarization. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Island of Maui: IEEE, pp. 10–20.

Brin, S., and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems* **30**(1): 107–117. Elsevier.

Buist, A. H., Kraaij, W., and Raaijmakers, S. 2005. Automatic summarization of meeting data: a feasibility study. In *Proceedings of the 15th CLIN Conference*, Leiden, The Netherlands.

Carbonell, J., and Goldstein, J. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia: ACM, pp. 335–336.

Carletta, J. *et al.* 2006. The ami meeting corpus: a pre-announcement. In *Proceedings of Machine Learning for Multimodal Interaction*, Brno, Czech Republic: Springer, pp. 28–39.

Chen, Y.-N., and Metze, F. 2012. Two-layer mutually reinforced random walk for improved multi-party meeting summarization. In *Proceedings of Spoken Language Technology Workshop*, Miami, Florida: IEEE, pp. 461–466.

Chen, Y. N., and Metze, F. 2013. Multi-layer mutually reinforced random walk with hidden parameters for improved multi-party meeting summarization. In *Proceedings of the Conference of the International Speech Communication Association(INTERSPEECH)*, Lyon, France, pp. 485–489.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science (JASIS)* **41**(6): 391–407.

Dielmann, A., and Renals, S. 2004. Dynamic bayesian networks for meeting structuring. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, Montreal, Canada. vol. 5, IEEE, pp. 626–629.

Eiben, A., and Smith, J. 2003. *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer, Springer-Verlag.

Engelbrecht, A. P. 2007. *Computational Intelligence: an Introduction.* England: John Wiley & Sons Ltd.

Erkan, G., and Radev, D. R. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)* **22**(1): 457–479. AI Access Foundation.

Fernández, R., Frampton, M., Ehlen, P., Purver, M., and Peters, S. 2008. Modelling and detecting decisions in multi-party dialogue. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, Columbus, OH, USA: ACL, pp. 156–163.

Fung, P., and Ngai, G. 2006. One story, one flow: hidden markov story models for multilingual multidocument summarization. *ACM Transactions on Speech and Language Processing (TSLP)* **3**(2): 1–16.

Galley, M. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia: ACL, pp. 364–372.

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

Garg, N., Favre, B., Riedhammer, K., and Hakkani-Tür, D. 2009. Clusterrank: a graph based method for meeting summarization. In *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*, Brighton, UK, pp. 1499–1502.

Georgescul, M., Clark, A., and Armstrong, S. 2009. An analysis of quantitative aspects in the evaluation of thematic segmentation algorithms. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, London, UK: ACL, pp. 144–151.

Germesin, S., and Wilson, T. 2009. Agreement detection in multiparty conversation. In *Proceedings of the International Conference on Multimodal Interfaces*, Cambridge, MA, USA: ACM, pp. 7–14.

Gillick, D., Riedhammer, K., Favre, B., and Hakkani-Tur, D. 2009. A global optimization framework for meeting summarization. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan: IEEE, pp. 4769–4772.

Goldberg, D. E., and Richardson, J. 1987. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms and their application*, Hillsdale, NJ, USA: Lawrence Erlbaum Associates, pp. 41–49.

Gong, Y., and Liu, X. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, Louisiana, USA: ACM, pp. 19–25.

Grosz, B. J., and Sidner, C. L. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* **12**(3): 175–204. MIT Press.

Hearst, M. A. 1997. Texttiling: segmenting text into multi-paragraph subtopic passages. *Computational Linguistics* **23**(1): 33–64. MIT Press.

Hillard, D., Ostendorf, M., and Shriberg, E. 2003. Detection of agreement vs. disagreement in meetings: Training with unlabeled data. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, Edmonton, Canada: ACL, pp. 34–36.

Hirohata, K., Okazaki, N., Ananiadou, S., Ishizuka, M., and Biocentre, M. I. 2008. Identifying sections in scientific abstracts using conditional random fields. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, Hyderabad, India, pp. 381–388.

Hirschberg, J., and Litman, D. 1993. Empirical studies on the disambiguation of cue phrases. *Computational Linguistics* **19**(3): 501–530. MIT Press.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, USA: ACM, pp. 50–57.

Holland, J. H. 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. *The University of Michigan Press*.

Hsueh, P. Y., Moore, J. D., and Renals, S. 2006. Automatic segmentation of multiparty dialogue. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy: ACL, pp. 273–280.

Joty, S. R., Carenini, G., and Ng, R. T. 2014. Topic segmentation and labeling in asynchronous conversations. *Journal of Artificial Intelligence Research* **47**: 521–573. AI Access Foundation.

Kong, S.-Y., and Lee, L.-S. 2006. Improved spoken document summarization using probabilistic latent semantic analysis (plsa). In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, Toulouse, France: IEEE, vol. 1, pp. 941–944.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *The Annals of Mathematical Statistics* **22**(1): 79–86. Institute of Mathematical Statistics.

Lamprier, S., Amghar, T., Levrat, B., and Saubion, F. 2007. SegGen: a genetic algorithm for linear text segmentation. In *Proceedings of the 20th International Joint Conferences on Artificial Intelligence (IJCAI)*, Hyderabad, India. vol. 7, pp. 1647–1652.

Lascarides, A., and Asher, N. 1993. Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and Philosophy* **16**(5): 437–493. Springer

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals.*Soviet Physics Doklady* **10**: 707.

Lin, C.-Y. 2004. Rouge: a package for automatic evaluation of summaries. In *Proceedings of the ACL Workshop*, Barcelona, Spain pp. 74–81.

Liu, F., and Liu, Y. 2008. Correlation between rouge and human evaluation of extractive meeting summaries. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, Columbus, OH, USA: ACL, pp. 201–204.

Liu, F., and Liu, Y. 2010. Exploring speaker characteristics for meeting summarization. In *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*, Makuhari, Japan, pp. 2518–2521.

Liu, Y., and Hakkani-Tür, D. 2011. Speech summarization. In *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, Wiley, pp. 357–396.

Liu, Y., Shriberg, E., Stolcke, A., Hillard, D., Ostendorf, M., and Harper, M. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on Audio, Speech, and Language Processing* **14**(5): 1526–1540. IEEE.

Luhn, H. P. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development* **2**(2): 159–165. IEEE.

Malioutov, I., and Barzilay, R. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia: ACL, pp. 25–32.

Mani, I., Klein, G., House, D., Hirschman, L., Firmin, T., and Sundheim, B. 2002. Summac: a text summarization evaluation. *Natural Language Engineering* **8**(1): 43–68. Cambridge University Press.

Mani, I., and Maybury, M. T. 1999. *Advances in Automatic Text Summarization*. Cambridge, MA: MIT Press.

Mann, W. C., and Thompson, S. A. 1988. Rhetorical structure theory: toward a functional theory of text organization. *Text Journal* **8**(3): 243–281. Australasian Association of Writing Programs.

Marcu, D. 1998. Improving summarization through rhetorical parsing tuning. In *Proceedings of the 6th Workshop on Very Large Corpora*, Montreal, Canada: ACL, pp. 206–215.

Maskey, S., and Hirschberg, J. 2003. Automatic summarization of broadcast news using structural features. In *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*, Geneva, Switzerland, pp. 1173–1176.

Maskey, S., and Hirschberg, J. 2005. Comparing lexical, acoustic/prosodic, structural and discourse features for speech summarization. In *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*, Lisbon, Portugal, pp. 621–624.

Maskey, S., and Hirschberg, J. 2006. Summarizing speech without text using hidden markov models. In *Proceedings of the Human Language Technology Conference of the NAACL*, New York city, USA: ACL, pp. 89–92.

McCowan, I., Bengio, S., Gatica-Perez, D., Lathoud, G., Monay, F., Moore, D., Wellner, P., and Bourlard, H. 2003. Modeling human interaction in meetings. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, Hong Kong; Hong Kong: IEEE, vol. 4, pp. 748–751.

McKnight, L., and Srinivasan, P. 2003. Categorization of sentence types in medical abstracts. In *Proceedings of the AMIA Annual Symposium*, American Medical Informatics Association, pp. 440–444.

Mehdad, Y., Carenini, G., Tompa, F. W., and NG, R. T. 2013. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*, Sofia, Bulgaria: ACL, pp. 136–146.

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

Murray, G., Renals, S., Carletta, J., and Moore, J. 2005. Evaluating automatic summaries of meeting recordings. In *Proceedings of the ACL MTSE Workshop*, Ann Arbor, MI, USA, pp. 33–40.

Murray, G., Renals, S., Carletta, J., and Moore, J. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, New York City, USA: ACL, pp. 367–374.

Murray, G., and Carenini, G. 2008. Summarizing spoken and written conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Honolulu, Hawaii. ACL, pp. 773–782.

Nenkova, A., and McKeown, K. 2012. A survey of text summarization techniques. In *Mining Text Data*, Springer, pp. 43–76.

Neto, J. L., Santos, A. D., Kaestner, C. A., and Freitas, A. A. 2000. Generating text summaries through the relative importance of topics. In *Advances in Artificial Intelligence*, Springer, pp. 300–309.

Oya, T., Mehdad, Y., Carenini, G., and Ng, R. T. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *Proceedings of the 8th International Natural Language Generation Conference*, Philadelphia, PA, USA. pp. 35–44.

Pevzner, L., and Hearst, M. A. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics* **28**(1): 19–36. MIT Press.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems* **14**(3): 130–137. Emerald Group Publishing.

Purver, M. 2011. Topic segmentation. In *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, Wiley, pp. 291–317.

Purver, M., Dowding, J., Niekrasz, J., Ehlen, P., Noorbaloochi, S., and Peters, S. 2007. Detecting and summarizing action items in multi-party dialogue. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, Belgium: ACL, pp. 200–211.

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

Raaijmakers, S., Truong, K., and Wilson, T. 2008. Multimodal subjectivity analysis of multiparty conversation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Honolulu, Hawaii: ACL, pp. 466–474.

Ramezani, M., and Feizi-Derakhshi, M. R. 2014. Automated text summarization: an overview. *Applied Artificial Intelligence* **28**(2): 178–215. Taylor and Francis.

Reiter, S., and Rigoll, G. 2004. Segmentation and classification of meeting events using multiple classifier fusion and dynamic programming. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, Cambridge, UK: IEEE, vol. 3, pp. 434–437.

Reiter, S., Schuller, B., and Rigoll, G. 2006. A combined lstm-rnn-hmm-approach for meeting event segmentation and recognition. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, Toulouse, France: IEEE, vol. 2, pp. 393–396.

Reiter, S., Schuller, B., and Rigoll, G. 2007. Hidden conditional random fields for meeting segmentation. In *Proceedings of the International Conference on Multimedia and Expo (ICME)*, Beijing, China: IEEE, pp. 639–642.

Reynar, J. C. 1994. An automatic method of finding topic boundaries. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL, pp. 331–333.

Riedhammer, K., Favre, B., and Hakkani-Tür, D. 2010. Long story short–global unsupervised models for keyphrase based meeting summarization. *Speech Communication* **52**(10): 801–815. Elsevier.

Salton, G., Singhal, A., Mitra, M., and Buckley, C. 1997. Automatic text structuring and summarization. *Information Processing and Management* **33**(2): 193–207. Elsevier.

Somasundaran, S., Ruppenhofer, J., and Wiebe, J. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, Belgium: ACL, pp. 26–34.

Wang, L., and Cardie, C. 2012. Focused meeting summarization via unsupervised relation extraction. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Seoul, South Korea: ACL, pp. 304–313.

Webber, B., Egg, M., and Kordoni, V. 2012. Discourse structure and language technology. *Natural Language Engineering* **18**(4): 437–490. Cambridge University Press.

Xie, S., Favre, B., Hakkani-Tür, D., and Liu, Y. 2009. Leveraging sentence weights in a concept-based optimization framework for extractive meeting summarization. In *Proceedings of 10th Conference of the International Speech Communication Association (INTERSPEECH)*, Brighton, UK, pp. 1503–1506.

Xie, S., and Liu, Y. 2008. Using corpus and knowledge-based similarity measure in maximum marginal relevance for meeting summarization. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, Las Vegas, NV: IEEE, pp. 4985–4988.

Xie, S., and Liu, Y. 2010. Improving supervised learning for meeting summarization using sampling and regression. *Computer Speech and Language* **24**(3): 495–514. Elsevier.

Xie, S., Liu, Y., and Lin, H. 2008. Evaluating the effectiveness of features and sampling in extractive meeting summarization. In *Spoken Language Technology Workshop*, Goa, India: IEEE, pp. 157–160.

Zechner, K. 2002a. Automatic summarization of open-domain multiparty dialogues in diverse genres. *Computational Linguistics* **28**(4): 447–485. MIT Press.

Zhang, D., Gatica-Perez, D., and Bengio, S. 2005. Semi-supervised meeting event recognition with adapted HMMs. In *IEEE International Conference on Multimedia and Expo*, Amsterdam, The Netherlands: IEEE, pp. 1102–1105.

Zhang, D., Gatica-Perez, D., Bengio, S., McCowan, I., and Lathoud, G. 2004. Multimodal group action clustering in meetings. In *Proceedings of the ACM 2nd International Workshop on Video Surveillance and Sensor Networks*, New York, NY, USA: ACM, pp. 54–62.

Zhang, J., Chan, R. H. Y., and Fung, P. 2007. Improving lecture speech summarization using rhetorical information. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding*, Kyoto, Japan: IEEE, pp. 195–200.

Zhang, J., and Fung, P. 2007. Speech summarization without lexical features for mandarin broadcast news. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, NY, USA: ACL, pp. 213–216.

Zhang, J., and Fung, P. 2012. Automatic parliamentary meeting minute generation using rhetorical structure modeling. *IEEE Transactions on Audio, Speech, and Language Processing*, **20**(9): 2492–2504. IEEE.