Generating Abstractive Summaries from Meeting Transcripts

Siddhartha Banerjee
The Pennsylvania State University
College of IST
PA, USA
sub253@ist.psu.edu

Prasenjit Mitra
Qatar Computing Research Institute
Hamad Bin Khalifa University
Doha, Qatar
pmitra@qf.org.qa

Kazunari Sugiyama
National University of Singapore
School of Computing
Singapore
sugiyama@comp.nus.edu.sg

ABSTRACT

Summaries of meetings are very important as they convey the essential content of discussions in a concise form. Both participants and non-participants are interested in the summaries of meetings to plan for their future work. Generally, it is time consuming to read and understand the whole documents. Therefore, summaries play an important role as the readers are interested in only the important context of discussions. In this work, we address the task of meeting document summarization. Automatic summarization systems on meeting conversations developed so far have been primarily extractive, resulting in unacceptable summaries that are hard to read. The extracted utterances contain disfluencies that affect the quality of the extractive summaries. To make summaries much more readable, we propose an approach to generating abstractive summaries by fusing important content from several utterances. We first separate meeting transcripts into various topic segments, and then identify the important utterances in each segment using a supervised learning approach. The important utterances are then combined together to generate a one-sentence summary. In the text generation step, the dependency parses of the utterances in each segment are combined together to create a directed graph. The most informative and well-formed sub-graph obtained by integer linear programming (ILP) is selected to generate a one-sentence summary for each topic segment. The ILP formulation reduces disfluencies by leveraging grammatical relations that are more prominent in non-conversational style of text, and therefore generates summaries that is comparable to human-written abstractive summaries. Experimental results show that our method can generate more informative summaries than the baselines. In addition, readability assessments by human judges as well as log-likelihood estimates obtained from the dependency parser show that our generated summaries are significantly readable and well-formed.

DocEng'15, September 8-11, 2015, Lausanne, Switzerland.
© 2015 ACM. ISBN 978-1-4503-3307-8/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2682571.2797061.

Table 1: Two sets of extractive summaries along with the corresponding gold standard human generated abstractive summaries from a meeting in the AMI corpus [5]. Set 2 follows Set 1 in the actual meeting transcript. "A," "B" and "D" refer to three distinct speakers in the meeting.

Set 1: Human-generated extractive summary
D: um as well as uh characters.
D: um different uh keypad styles and s symbols.
D: Well right away I'm wondering if there's um th th uh, like with DVD players, if there are zones.
A: Cause you have more complicated characters like European languages, then you need more buttons.
D: I'm thinking the price might might appeal to a certain market in one region, whereas in another it'll be different, so
D: kay trendy probably means something other than just basic
Abstractive summary: The team then discussed various features to consider in making the remote.
Set 2: Human-generated extractive summary
B: Like how much does, you know, a remote control cost.
B: Well twenty five Euro, I mean that's um that's about like eighteen pounds or something.
D: This is this gonna to be like the premium product kinda thing or
B: So I don't know how how good a remote control that would get you. Um.
Abstractive summary: The project manager talked about the project finances and selling prices.

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: Natural Language Processing—*Language generation*

Keywords

Abstractive meeting summarization; Integer linear programming; Topic segmentation

1. INTRODUCTION

Meeting summarization helps both participants and non-participants by providing a short and concise snapshot of the most important content discussed in the meetings. While previous work on meeting summarization was primarily extractive [14, 15], a recent study showed that people generally prefer abstractive summaries [29].

Table 1 shows the human-written abstractive summaries along with the human-generated extractive summaries from the AMI cor-

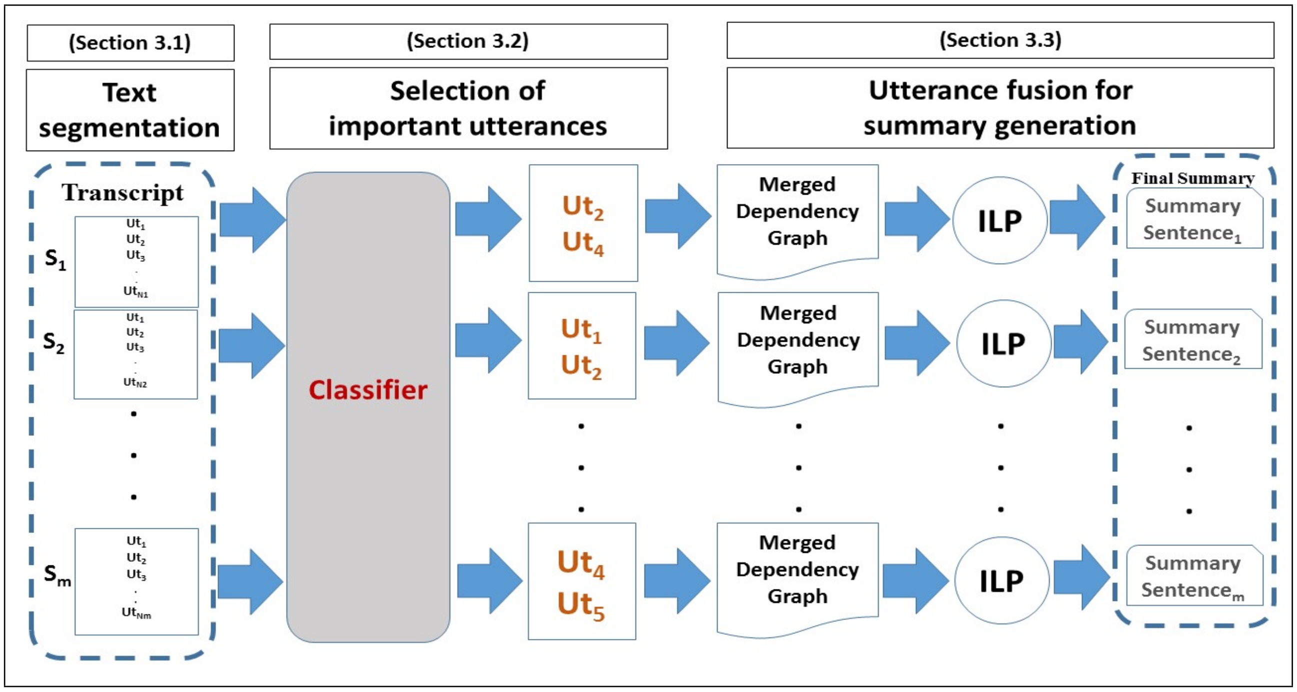

Figure 1: Our meeting summarization system overview.

pus [5]. Set 1 and Set 2 show two different topics discussed in the meeting – *design features* and *finances*. We have skipped other intervening utterances not included in the extractive summary. As shown in Table 1, the utterances are highly noisy and contain unnecessary information. Even if an extractive summarizer can accurately classify these utterances as "important" and generate a summary, it is usually hard to read and synthesize information from such summaries. In contrast, the human written summaries are compact and written in non-conversational style. They are more readable than the extractive summaries and preserve the most important information.

Previous approaches to abstractive meeting summarization have relied on template-based [36] or word-graph fusion-based [26] methods. The template-based method was applied to the generation of *focused summaries*.[1] Template-based generation is feasible in the case where the type of the summary is known apriori; however, our work does not make any assumptions on the type of the summary to be generated. The word-graph fusion-based technique, on the contrary, used an unsupervised approach to fuse a cluster of utterances generated using an entailment graph-based approach. However, this method did not take into consideration any grammatical dependencies between the words, resulting in ungrammatical output in several cases.

In this work, we propose an automatic way of generating short and concise abstractive summaries of meetings. Every meeting is usually comprised of several sub-topics [19]. As shown in Table 1, the participants discuss different aspects in Set 1 and Set 2. A well-formed abstractive summary should identify the most important aspects discussed in the meeting. In other words, if we can summarize the important information from every aspect, we can generate an informative summary that highlights the salient elements of the meeting. Therefore, we need to determine the boundaries where significant topic changes happen to isolate different aspects. Further, to generate a summary for each segment (topic), we should be able to fuse information from multiple utterances on that topic and retain the most informative constituents. Simultaneously, we should also generate grammatical output to ensure that the final summaries are well-formed and readable.

[1]Focused summary refers to summaries on specific aspects of the meeting such as actions, decisions, etc.

Figure 1 shows the overview of our proposed meeting summarization system. As shown in Figure 1, initially, a meeting transcript is divided into several topic segments S_i $(i = 1, 2, \ldots, m)$, where each segment contains N_i utterances $(Ut_{N_1}, \ldots, Ut_{N_i})$ on a specific topic. Previous work on meeting summarization [28] has shown that *lexical cohesion* is an important indicator in topic identification in meetings. We experiment with two different lexical-cohesion based text segmentation algorithms: LCSeg [13] and unsupervised Bayesian topic segmentation [8]. Only a few utterances contain information that is worthy of being included in the summary. Therefore, we introduce an extractive summarization component. To identify the most important (summary-worthy) utterances, we employ a supervised learning approach to construct a classifier by using content and discourse-level features. We parse the important utterances in each segment using a dependency parser, and then fuse the corresponding dependency graphs together to form a directed graph (*merged dependency graph*). The directed graph consists of the words in the utterances as the nodes, while the edges represent the grammatical relations between the words. Such a graph construction method ensures fusion of common information elements from utterances within the same topic segment. We introduce an *anaphora resolution* step when merging dependency graphs. We also introduce an *ambiguity resolver* that takes into consideration the context of words when fusing several utterances. Consider the following two utterances:

"um there's a sample sensor and there's a sample speaker unit",

"I'm not sure how the sample unit gonna work."

Once the first utterance is added into the graph, two nodes containing the word "*sample*" are created. The ambiguity resolver maps the word "*sample*" from the second utterance to the second node ("*sample*" node adjacent to "*unit*") to account for the correct context of the words. Our goal is to retain the most informative nodes (words) in the graph. Further, linguistically well-formed grammatical relations should be retained. We formulate the sub-graph generation problem as an Integer Linear Programming (ILP) problem by adapting an existing sentence fusion technique [10]. The solution to the ILP problem generates a sub-graph that satisfies several constraints to maximize information content and linguistic quality. Information content is measured using Hori and Furui's word informativeness formula [18] while the linguistic quality is estimated using probabilities of grammatical relations from the Reuter's corpus [1]. Grammatical relations extracted from the Reuter's corpus assign higher preferences to non-conversational style of text, thereby resulting in summaries that mirror the flair of human-written abstracts. In the ILP problem, we introduce constraints to limit the length of the sentences. Further, we ensure connectivity in the graph. We introduce several linguistic constraints to generate grammatical output. The sub-graph generated from each segment is linearized [11] using a bottom-up approach to generate a one-sentence summary. The one-sentence summaries from all the segments are combined in the final summary. Note that we do not introduce any new phrases or words in the process of combining information from multiple utterances. Instead, we consider utterances that are associated with the same topic and apply the ILP-based fusion technique to identify grammatical relations that contains more informative phrases, at the same time leading to generation of summaries that are fairly readable.

To the best of our knowledge, this is the first work that addresses the problems of readability, grammaticality, and content selection jointly for meeting summary generation without employing a template-based approach. Experimental results on the aforementioned AMI corpus that consists of meeting recordings show that our approach outperforms the comparable systems. ROUGE-2 and ROUGE-SU4 [20] scores from our abstractive model (0.048 and 0.087) are significantly better than that of the extractive summaries (0.026 and 0.044) as well as the word-graph based abstractive summarization method [26] (0.041 and 0.079). We also assess readability of the summaries using a human judge, demonstrating that the summaries generated by our method are fairly well-formed.

2. RELATED WORK

In the field of meeting summarization, while extractive techniques have been widely employed so far [22, 24], abstractive techniques, including sentence compression, template and graph-based approaches, have been focused on recently.

Liu and Liu [23] used sentence compression to generate summaries of meetings. However, they reported that the quality of the generated summaries are not so good and there is a potential limit to apply such methods to summarization. Murray *et al.* [30] mapped conversations to an ontology that was complemented with a Natural language generation (NLG) component used for transforming utterances to summaries. The corresponding full summarization system was later presented in [29], where a user study was conducted on the abstractive summaries that were generated. However, the full summarization system involved extensive manual labor to set specific speakers, entities, etc. in a template before using an NLG realizer to generate the summaries. Lu and Cardie [36] proposed a method that learns templates from the human written summaries and generates the summaries of decisions and actions of meetings by using the best set of templates for a particular summary ranked using a greedy approach. In contrast, we cannot use templates because we assume that the type of a conversation (action, decision, etc) is not known apriori.

Mehdad *et al.* [26] developed a method that first over-generates multiple fused utterances in an entailment graph, and then chooses one based on the final path ranking. The fusing of the utterances only considers words, and ignores the grammatical relations between them. This results in generation of summaries with poor linguistic quality. More recently, Oya *et al.* [32] used the same fusion technique to generate summaries of meetings. Both of the methods developed by Mehdad *et al.* and Oya *et al.* mentioned above relied on using multi-sentence compression (MSC) [9] that combines information from sentences that are similar or connected using some common entity. The MSC technique is a word-graph based method where multiple sentences or utterances can be represented as a network of words. A directed graph is generated where the nodes represent the words while edges exist if two words are adjacent in the utterances. From the graph, several paths between the start and end points can be generated. The new paths can represent content that can be different from the original utterances. Oya *et al.*'s proposed approach requires significant effort to generate the templates using hypernym information for creating slots in the templates. Our framework also consists of a similar segmentation module as employed in Oya *et al.*'s work, which ensures that we divide the meeting transcript into several topics. Our proposed method is fundamentally different from most of the aforementioned techniques (except Oya *et al.*'s work) in that it considers individual segments to generate a summary sentence.

Our previous work [2] has briefly described the effectiveness of the fusion-based technique, which is also employed in this work. Our preliminary results demonstrated that the fusion based model can combine and convey useful information, generating reasonable abstractive meeting summaries. Hence, we extend this work using topic segments to build an end-to-end framework. We address the issue of readability of the generated summaries by modeling the strength of grammatical relations in the optimization problem. Our approach does not require creation of templates. Instead, our model aims to generate a sentence on each topic by identifying relevant grammatical relations and informative words from a collection of important utterances in a meeting.

3. PROPOSED APPROACH

As explained in Section 1, our proposed approach consists of three steps: First, we segment an entire conversation between participants into multiple text segments. Second, we apply an extractive summarizer that extracts important utterances from each segment. Finally, we fuse all the utterances in a segment using an ILP based approach to generate a summary sentence. All the generated sentences are appended to create the final summary. In the following, we detail each step.

3.1 Text Segmentation

Topic segmentation has been used in summarization of news articles [21, 4]. Generally, lexical cohesion-based measures work well for topic segmentation [27]. As the primary focus of our work is to generate summaries, we experiment with two different text segmentation algorithms: **LCSeg** and **Bayesian unsupervised topic segmentation**.

LCSeg: Galley *et al.* [13] developed a topic segmentor, LCSeg, based on lexical cohesion, which is considered to be a good indicator of the discourse structure of the text. The intuition behind this algorithm is that major term repetitions occur when the underlying topics in the text start or end. It takes into consideration multiple

Table 2: Features to select important utterances. Most of them are adopted from previous works [12, 37]. The most important speaker refers to the one that utters maximum number of words. Our work introduces the segment-based features. The content words include nouns, adjectives, verbs and adverbs.

(1)	**Basic features**
	– Length of a dialogue
	– Number of content words
	– Portion of content words
	– Number of new nouns introduced
(2)	**Content features**
	– Cosine similarity with entire meeting transcript
	– Presence of proper nouns
	– Most important speaker in meeting
	– Content words in previous dialogue act
(3)	**Segment based features**
	– Most important speaker in segment
	– Cosine similarity of dialogue with entire segment

features such as discourse cues and overlaps. LCSeg is applied to meeting corpora and achieved promising results. Hence, this approach is suitable for our segmentation step.

Bayesian unsupervised topic segmentation: This is also promising approach to topic segmentation. Eisenstein and Barzilay [8] proposed an unsupervised approach to topic segmentation based on lexical cohesion modeled by a Bayesian framework. The cohesion arises through a generative process. The words are modeled from a multinomial language model and the observed likelihood is maximized to generate a lexically-cohesive segmentation. This algorithm requires a user to specify the desired number of segments.

3.2 Selection of Important Utterances

Our second step is to identify the set of important utterances in each topic segment. As shown in Table 2, we use multiple features to identify the important set of utterances in a meeting. We adopt basic and content features from previous works [12, 37]. In addition to the above mentioned features, we introduce two segment-based features:

(i) The most important speaker in a segment.

(ii) Cosine similarity between the utterance and all of the other utterances in a segment.

We construct classifiers using all the features on the training set. We conduct experiments to evaluate the impact of our introduced segment-based features in addition to the basic and content features. Moreover, constructing a model using the meeting summarization data also suffers from the unbalanced data problem as only few utterances are considered to be important to generate the final summary. In order to address this problem, we apply the following techniques to oversample the minority data:

Weight: Let $npRatio$ be $\frac{\#negative}{\#positive}$. For the training instances, we assign weights of one and $npRatio$ to the negative and positive examples, respectively.

Resampling: We reproduce a random subsample of the training data using sampling with replacement. In this case, the new training data has the same total number of samples as the old one. However, they contain equal populations in both of the classes.

SMOTE: In synthetic minority oversampling technique (SMOTE) [6], the minority class is randomly oversampled. This algorithm forms new examples of minority class by interpolating between several minority class examples that lie together and thereby can avoid the overfitting problem.

3.3 Fusion of Utterances for Summary Generation

The final step in our approach is to combine information from multiple extracted utterances in each segment that the classifier identifies as summary-worthy. Several techniques have been proposed for sentence fusion tasks [3]. However, fusion on meeting utterances requires an algorithm that is robust for noisy data as utterances often have disfluencies. We adapt a sentence fusion technique [10] to meeting utterances. The dependency parse trees of the individual utterances within a topic segment are combined together. The best sub-graph that satisfies several constraints and maximizes the propagated information is selected using as an integer linear programming (ILP) formulation. ILP has been applied successfully to many natural language processing tasks [7, 34]. The formulation of the objective function in the ILP function takes into consideration the informativeness of the words, weights of the edges along the dependency tree and a factor that assigns more weights to utterances that are more closer to topic shifts, *i.e.*, towards the end of a segment. We also introduce an additional step of pronoun resolution. We observe that a lot of pronominal references are used in utterances and resolving such references would produce more relevant fusion by merging dependency graphs. Finally, the solution of the ILP problem is linearized to produce a sentence. In this section, we explain all of the details using a simple example. Suppose that the following three utterances within a topic segment are labeled as important by the classifier:

(Ut_1) *"Um well this is the kick-off meeting for our project."*

(Ut_2) *"so we're designing a new remote control and um."*

(Ut_3) *"Um, as you can see it's supposed to be original, trendy and user friendly."*

As can be seen, there are the introductory statements in a meeting that discusses the purpose of the meeting. We apply pre-processing to get rid of words such as "*um*," "*ah*" that cause disfluencies and do not contribute to any information content in the utterances.

Anaphora resolution. Our final goal is to generate a one-sentence summary from these utterances. To obtain a summary for each segment, we fuse the dependency graphs of the utterances by merging them on the common words that represent the nodes in the graph. However, in the above example, there are no common words in the three utterances. As can be seen from the utterances, the "*it*" in utterance (Ut_3) refers to "a new remote control" in (Ut_2). To ensure accurate dependency graph merging, where the graphs are merged on the nodes (words in an utterance), it is important to resolve such pronominal references. Without resolving such references, it would be impossible to fuse the above utterances, even though they are referring to the same entity. We use the publicly available Stanford CoreNLP package[2] [25] that has a co-reference resolution module. We resolve pronouns only if there is a pronominal reference to the previous utterance.

Dependency graph merging. Once anaphora resolution has been applied, the extracted utterances in each segment are parsed using

[2] http://nlp.stanford.edu/software/corenlp.shtml

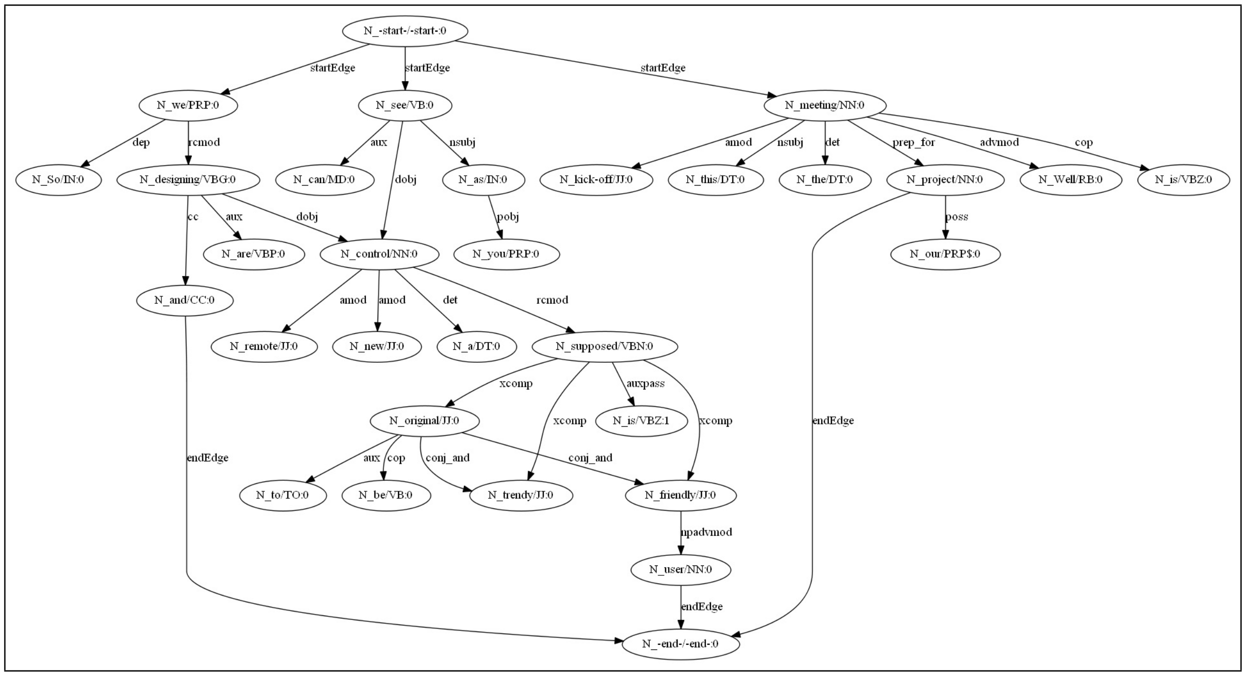

Figure 2: A merged dependency graph generated from several utterances. Note that only a section of the entire graph is shown. The nodes are shown as "*N_word*" and the labels are placed on the edges.

the Stanford dependency parser that is also a part of the CoreNLP package. Every individual utterance has an explicit ROOT vertex. We add two dummy nodes in the graph: the *start* node and the *end* node. The ROOT nodes from the utterances are all connected to the *start* node and the last word of every utterance is connected to the *end* node. The words from the utterances are iteratively added onto the graph. The words that have the same word form and the parts of speech (POS) tag are assigned to the same nodes. While only content words are merged, stopwords are not merged. The use of POS information prevents ungrammatical mappings. Hereafter, we refer to a word as the tuple of "{word, POS}." We also address ambiguities in the word mappings. If a new word that needs to be merged onto the graph has multiple mapping candidates, we introduce an ambiguity resolver.

Ambiguity resolver. Suppose that a new word w_i that has k ambiguous nodes where it can be mapped to. The k ambiguous nodes are referred to as mappable nodes. For every ambiguous mapping candidate, we first find the words to the left and right of the mappable word of the sentences, and then compute the number of words in both of the directions that are common to the words in either direction of the word w_i. We define the directed context as follows:

$$dirContext = \#CommonWords(dir, window),$$

where dir and $window$ denote the direction of context (*left*/*right*) and the number of words to be considered in either direction, respectively. We calculate the directed context in both of the directions upto a window size of two words. Finally, w_i is mapped to the node that has the highest directed context. If a tie cannot be broken or no common context can be found with any of the existing nodes, a new node for w_i is created. An example of the ambiguity resolution has been provided in Section 1.

We use the JGrapht[3] package for the generation of the graph structure. Figure 2 shows a snapshot of the merged dependency graph generated from the three utterances, Ut_1, Ut_2, and Ut_3 in Section 3.3. The three utterances have been combined together in a common structure, with various possible paths between the *start* and the *end* dummy nodes. To obtain the dependency relations, we use the "collapsed dependency representation" from the Stanford parser that collapses edges of conjunctions and prepositions and places the corresponding information on the edge labels (*e.g.*, conj_and, prep_at, etc).

ILP formulation. The next step is to solve and generate a subgraph from this structure that satisfies a number of syntactic constraints and maximizes the information content simultaneously.

Similar to the fusion technique by Fillipova and Strube [10], we model the problem as an integer linear programming (ILP) formulation. However, the formulation of our objective function and the constraints are significantly different from their system. We add a lexical cohesion component in the ILP formulation. Moreover, our constraints leverage linguistic knowledge to generate grammatical output. Furthermore, they applied it to German language only. The directed edges in the graph are represented as $x_{g,d,l}$ in the ILP problem where g, d and l denote the governor node, dependent node and the label of an edge, respectively. The edges represent the variables in the objective function which can either take value of 1 or 0 depending on whether the edge has to be preserved or deleted.

We maximize the following objective function:

$$\sum\nolimits_x x_{g,d,l} \cdot p(l \mid g) \cdot I(d) \cdot \frac{p_x}{N}. \tag{1}$$

As shown in Equation (1), we introduce three different terms: $p(l \mid g)$, $I(d)$ and $\frac{p_x}{N}$. The term $p(l \mid g)$ denote the probabilities of the labels given a governor node, g. We can calculate these probabilities from any given corpora. For every node (word and POS)

[3]`http://jgrapht.org/`

Table 3: Probabilities of outgoing edges from a node for "*produced/VBN*."

aux-pass	nsubj-pass	aux	prep_with	agent	prep_in	adv-mod
0.286	0.214	0.214	0.071	0.071	0.071	0.071

in the entire corpus, the probabilities are represented as the ratio of the sum of the frequency of a particular label and the sum of the frequencies of all the labels emerging from a node. In this work, we calculate these values using Reuters corpora [33] in order to obtain dominant relations from non-conversational style of text. For example, Table 3 shows the probabilities of outgoing edges from a node ("*produced/VBN*"). The term $I(d)$ denotes the informativeness of a node. In order to compute $I(d)$, we improve the word significance score [18] as follows:

$$I(d) = f_s \cdot \log \frac{F_A}{F_d}, \tag{2}$$

where f_s, F_A, and F_d denote the frequency of a word in a text segment, the sum of the frequencies of all the words in the corpus, and the frequency of the dependent word d in the entire Reuters corpus, respectively. The last term $\frac{p_x}{N}$ in Equation (1) is based on the idea of lexical cohesion. Our intuition is that important decisions in a meeting are taken just before a topic concludes. Therefore, to model the relative importance of such utterances, we introduce the term $\frac{p_x}{N}$, where N and p_x denote the total number of extracted utterances in a segment and the position of the utterance (the edge x belongs to) in the set of N utterances, respectively. As a result of this term, utterances more closer to topic boundaries are assigned higher weights.

In order to solve the above ILP problem, we impose a number of constraints. Some of the constraints have been directly adapted from the original ILP formulation [10]. For example, we use the same constraints for restricting one incoming edge per node, as well as we impose the connectivity constraint to ensure a connected graph structure. The other constraints we impose are defined as follows:

$$\begin{gathered} \forall l \in startEdge, \sum\nolimits_l x_{g,d,l} = 1, \\ \forall l \in endEdge, \sum\nolimits_l x_{g,d,l} = 1 \end{gathered} \tag{3}$$

$$\sum\nolimits_x x_{g,d,l} \leq \gamma \tag{4}$$

$$\sum\nolimits_{g,d} (x_{g,d,l} + x_{d,g,l}) \leq 1 \tag{5}$$

$$\forall l_{out} \in \{aux, cop, det\}, \sum\nolimits_{u,l_{in}} x_{g,u,l_{in}} - x_{u,d,l_{out}} = 0 \tag{6}$$

$$\forall g, l_{out} \in aux \vee cop \vee det, \sum\nolimits_{l_{out}} x_{g,d,l_{out}} \leq 1 \tag{7}$$

Equation (3) limits the subtree to compulsorily have just one start edge and one end edge. This helps in preserving one ROOT node, as well as it limits to one *end* node for the generated subtree. Equation (4) limits the generated subtree to have a maximum of γ nodes. The start nodes and end nodes are still a part of the subtree that is generated by solving this optimization problem. Hence, the value of γ needs to be set to 2, which is more than the desired number of maximum words in the summary sentence. In order to prevent bidirectional relations between two nodes, we impose Equation (5) as a constraint. To maintain the linguistic quality of the generated sentence, by using Equations (6) and (7) as constraints, we always include a maximum of one auxiliary verb (aux), copular verb (cop) and determinant (det) if they exist. We use the Gurobi software[4] [16] for the optimization tasks. Figure 3 shows the final graph that is retained from the graph in Figure 2 after solving the ILP problem.

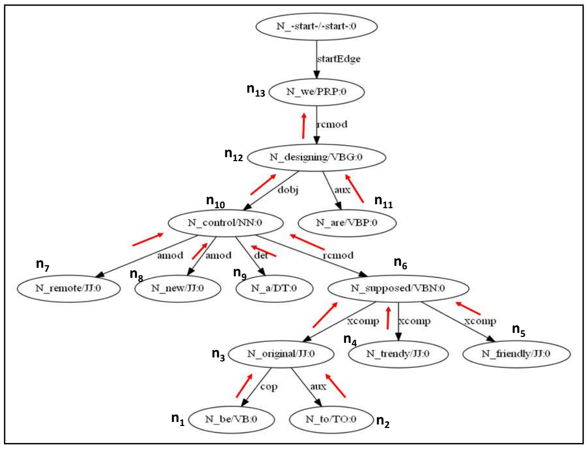

Figure 3: Dependency graph linearization process.

Linearization: The purpose of linearization is to generate a sentence from the final subtree generated by solving the ILP problem. We take a relatively straightforward bottom-up approach to tackle the problem. We keep adding the leaf nodes to their governing nodes until it reaches the ROOT node. We maintain the same order of the words as in the source sentences during the merging process.

As shown in Figure 3, the nodes *to/TO (n_2)* and *be/VB (n_1)* are added to the node *original/JJ (n_3)*. After merging these words, they are reordered so that the ordering resembles the one in the source sentences. The sequence of the nodes is changed only during the merge with the governing node: they are kept fixed for the future operations. Hence, the nodes *to/TO (n_2)* and *be/VB (n_1)* are memorized along with the node *original/JJ (n_3)*. In the next step, the ordering of the node *original/JJ (n_3)* matters with respect to the other leaves the governing node (n_6) has. However, this might be a problem in the case where there are leaf nodes from a governor node at some higher level. In this example, *trendy/JJ (n_4)* and *friendly/JJ (n_5)* will get merged to *supposed/VBN (n_6)* before the node *original/JJ (n_3)* as they are leaf nodes. To prevent such merging, we only allow to merge the leaf nodes to the governing node that is at the farthest distance from the ROOT vertex. We apply Dijkstra's algorithm [35] to calculate the path length. Thus, in Figure 3, the nodes *trendy/JJ (n_4)* and *friendly/JJ (n_5)* are added only after *to/TO (n_2)* and *be/VB (n_1)* are merged to *original/JJ*. The final sentence after linearization is as follows:

> *We are designing a new remote control supposed to be original trendy and friendly.*

4. EXPERIMENTAL RESULTS

4.1 Dataset and Evaluation Metrics

The AMI Meeting corpus [5] contains 139 meeting transcripts along with their corresponding extractive and abstractive summaries. The standard test set of this corpus includes 20 meetings. Our extractive summarization component is trained using the training set,

[4]`http://www.gurobi.com/`

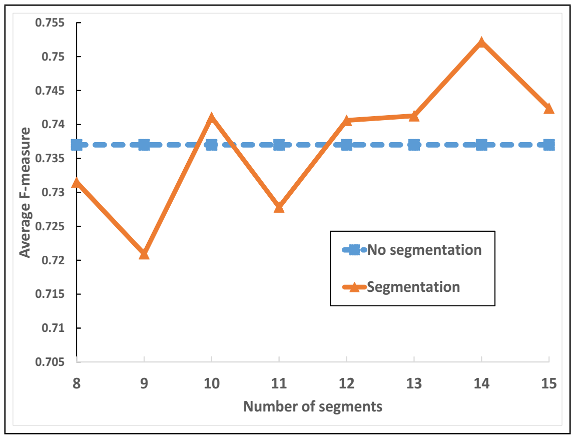

Figure 4: Average F-measures obtained by varying the number of segments. Also shows the impact of addition of segment-based features.

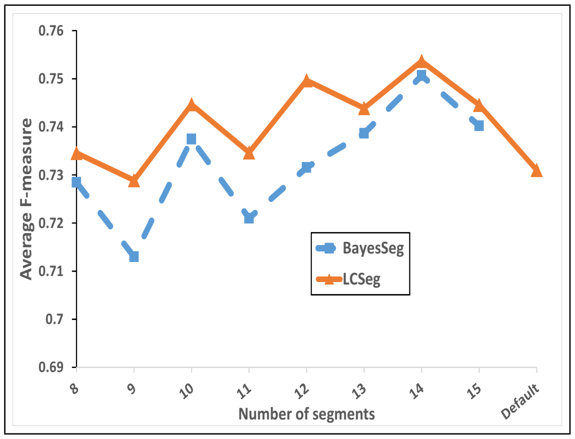

Figure 5: Comparison of performance obtained by text segmentation algorithms: Bayesian segmentation (BayesSeg) and LCSeg. "Default" setting does not require the number of segments to be explicitly stated when computing segment boundaries using LCSeg.

Table 4: Precision (Pre), Recall (Rec), and F-measure (F) obtained by classifiers along with each of the three sampling strategies. We set the number of segments to 14.

		LCSeg			BayesSeg		
Classifier	**Sampling**	**Pre**	**Rec**	**F**	**Pre**	**Rec**	**F**
NB	Weight	0.792	0.787	0.786	0.788	0.784	0.783
	Resampling	0.796	0.790	0.789	0.791	0.787	0.786
	SMOTE	0.831	0.804	0.811	0.830	0.807	0.813
RF	Weight	0.726	0.722	0.721	0.730	0.727	0.727
	Resampling	0.892	0.889	**0.888**	0.880	0.877	**0.877**
	SMOTE	0.817	0.821	0.819	0.817	0.820	0.818
SVM	Weight	0.623	0.591	0.563	0.601	0.584	0.566
	Resampling	0.752	0.741	0.739	0.690	0.689	0.689
	SMOTE	0.660	0.694	0.667	0.691	0.711	0.697

i.e., the remaining 119 meetings. We evaluate the accuracy of the classifiers using standard metrics: Precision, Recall and F-measure. We also evaluate the impact of introducing segment-based features. To evaluate the quality of the summaries, we verify the effectiveness of content selection using ROUGE, which has been widely used as a standard technique to evaluate information content in document summarization tasks by comparing system-generated summaries to human-written abstractive summaries. Further, we also evaluate the linguistic quality of the generated summaries using human judgments.

4.2 Classifier Evaluation

As described in Section 3.1, we used two different text segmentation algorithms: LCSeg and Bayesian unsupervised topic segmentation. Furthermore, we employed three classifiers: Support Vector Machines (SVM), Random Forest (RF) and Naive Bayes (NB). To overcome the problem of unbalanced data, we used three different sampling techniques (Weight, Resampling, and SMOTE) as described in Section 3.2. We evaluated all the possible configurations on the training dataset to determine the best configuration suitable for our summary generation. We used Weka [17] for all the classification tasks with the default set of parameters. We perform 10-fold cross validation on the training set. First, we try to find the optimal number of segments that provides the best classification accuracy. Simultaneously, we also evaluate the contribution of adding segment-based features during training the classifiers. Second, we also identify the text segmentation algorithm that works best on this dataset. Finally, based on the above decisions, we compare the performances of the classifiers to decide our extractive summarization component (*i.e.*, classifier that gives the the best) and the best sampling strategy to avoid any bias due to the unbalanced dataset.

Number of segments: We optimize the number of segments for each meeting by varying it from 8 to 15 on the training data. Figure 4 shows the average F-measure obtained by the classifiers constructed from the set of all features ("Segmentation") and the features excluding the segment-based features ("No segmentation"). The graph shows the average F-measure obtained by all combinations of the classifiers and the sampling strategies with respect to the various number of segments. As expected, the average F-measures do not show any change when we skip the segment-based features. However, we observe slight differences when we introduce the segment-based features. Generally, when the number of segments is between 12 to 15, we observe about 1% improvement in F-measure by adding the segment-based features over the basic set of features. According to Figure 4, we observe the highest average F-measure of 0.752 when we segment the meeting transcript into 14 topics. We set the number of segments to 14 for our following experiments.

LCSeg vs Bayesian unsupervised topic segmentation: Figure 5 shows the comparison of average F-measures of the classification models for each of the topic segmentation algorithms. As can be seen, LCSeg generally outperforms Bayesian segmentation. The *default* setting of the number of segments refers to the setting for LCSeg, which does not require to specifying the number of text segments. According to Figure 5, the default setting does not perform well. Both of the segmentation algorithms achieve similar F-measure in classification accuracies when we set the number of segments to 14. Note that the optimal number of segments were obtained by applying 10-fold cross validation using both the segmentation algorithms.

Evaluation of classifiers and sampling strategy: Table 4 shows the results of classification evaluation. The scores in the table were obtained by setting the number of segments to 14. As can be seen from the table, the NB and RF classifiers significantly outperform SVM. The best system is obtained by combining the RF classifier with the Resampling strategy. In both of the segmentation algorithms, the combination of RF and Resampling gives the best F-measure (0.888 and 0.877). Resampling and SMOTE sampling

Table 5: ROUGE scores obtained by several configurations in content selection. We compute ROUGE-1 (R-1) and ROUGE-2 (R-2) without any limit on summary length for comparison.

		LCSeg		BayesSeg	
Classifier	**Sampling**	**R-1**	**R-2**	**R-1**	**R-2**
NB	Weight	0.660	0.141	0.663	0.142
	Resampling	0.666	0.142	0.674	0.145
	SMOTE	0.673	0.145	0.675	0.147
RF	Weight	0.679	0.147	0.702	0.144
	Resampling	**0.705**	**0.158**	**0.703**	**0.157**
	SMOTE	0.694	0.152	0.695	0.153
SVM	Weight	0.490	0.114	0.473	0.112
	Resampling	0.563	0.143	0.556	0.139
	SMOTE	0.525	0.123	0.567	0.141

Table 6: Content selection evaluation. We compute ROUGE-2 (R-2) and ROUGE-SU4 (R-SU4) scores by comparing the system generated summaries and the human-written summaries for all the meetings in the test set.

Method	R-2	R-SU4
Our abstractive model	**0.048**	**0.087**
Our abstractive model (no anaphora resolution)	0.036	0.071
MSC model [9]	0.041	0.079
Extractive model (baseline)	0.026	0.044

strategies outperform the Weight strategy when using RF and SVM. However, when using NB, the performance using Weight and Resampling strategy is very similar.

4.3 Content Selection

In text summarization, it is also important to evaluate to what extent a classifier retains valuable information that should exist in a summary. Therefore, system generated summaries should be compared to human-written summaries automatically. We evaluate content selection using ROUGE.

Training Set: Table 5 shows the experimental results of content selection on the training set. We compare extractive summaries with human-generated abstracts for all the meetings in the training set and compute ROUGE-1 (R-1) and ROUGE-2 (R-2) scores. We do not impose any length constraints during ROUGE evaluation on the training data. Similar to Table 4, the combination of RF with Resampling strategy outperforms other techniques in terms of R-1 and R-2 scores. We use the RF classifier trained using Resampling strategy as our extractive component. This classifier is used on the test set to identify important utterances in the meeting transcripts. We segment each meeting transcript into 14 (optimal) segments using the LCSeg algorithm as it slightly outperform Bayesian unsupervised topic segmentation.

Test-set evaluation: We generate abstractive summaries from the meeting transcripts in the test set using our ILP-based approach. To evaluate our abstractive summaries, we compare them to the extractive summaries generated by the best performing classifier. We also compare the summaries to the MSC method proposed by Fillipova [9] that has been adapted for abstractive meeting summarization [26] develped earlier. As an input to the MSC model, we use the same set of utterances per segment that was extracted by the classifier. The sentence in each segment that obtains the highest score using MSC is used in the final generated summary. The human-written abstracts, on average, contain close to 300 words. Therefore, we applied a length constraint while performing ROUGE evaluation to limit summary comparison upto 300 words.

Table 7: Readability estimates of summaries.

Method	Readability score	Log likelihood
Our abstractive model	0.74	**-125.73**
MSC model [9]	0.62	-141.31
Extractive model	0.67	-136.22

We use ROUGE-2 (R-2) and ROUGE-SU4 (R-SU4) recall scores to compare all approaches. Both the ROUGE scores have been found to correlate well with human judgments [31]. Table 6 shows that our abstractive model can effectively maximize information content, resulting in better summaries compared with the other models. Furthermore, the ROUGE scores of the MSC model also significantly outperforms the extractive model, indicating that MSC results in more informative summaries. To evaluate the impact of anaphora resolution, we run our abstractive summarization model without performing the pre-processing step of pronoun resolution. ROUGE-2 score obtained by the abstractive model with anaphora resolution (0.048) is significantly better than the model without anaphora resolution (0.036). This indicates that anaphora resolution significantly contributes to content selection. Due to pronoun resolution, there are more chances of fusing information from various utterances within a topic segment. The extractive model obtains the lowest ROUGE scores as we restrict comparison to the first 300 words. In contrast, the abstractive methods (our methods and MSC) can effectively integrate the information from multiple utterances within the first 300 words.

ROUGE comparison: In general, shorter summaries are preferred by human readers. Extractive meeting summaries tend to be very long. In contrast, human-written summaries are very short and contain 300 words on average. To take this preference of shorter summaries into account, we evaluate summaries using only the first 300 words. The extractive summaries are fairly long and contain 2000-5000 words. Including several utterances distracts a reader's focus on the salient aspects, resulting in low readability of the summaries. Therefore, we set the length parameter in ROUGE (l) to 300 to limit the comparison to only the first 300 words.

4.4 Readability Analysis

To evaluate the linguistic quality of the generated summaries, we perform readability analysis. We asked one human judge to mark sentences in the generated abstractive summaries as either readable or not readable. Readability indicates how well the idea in the sentence is conveyed to the reader. Excessive presence of disfluencies or ill-formed utterances should be marked as not readable. We provided these instructions to the human judge. Out of 261 summary sentences generated for the 20 abstractive summaries, 67 sentences were found to be *not readable* ($\sim$26%). We also manually evaluated the extracted utterances and found that 33% of the utterances contained various disfluencies, making them difficult to read. We also performed another readability analysis for the summaries generated using MSC and found that only 62% of the generated sentences in the summaries are readable. The readability of MSC summaries (0.62) is even worse than that of the extractive summaries (0.67), showing that, while the generated sentences in MSC model are informative (high ROUGE scores), they suffer from serious grammatical issues as no factor of linguistic quality is considered in the model. Generally, several utterances in extractive summaries were marked as not readable due to excessive use of disfluencies. Furthermore, there were incomplete utterances that created confusion in the minds of the reader. For example, an extracted utterances – "Ah eagle , right okay ." – although gram-

Table 8: Examples of summary sentences that our system generated (S) and corresponding human–written (H) summary sentences. Note that they are only small portions of the summaries and not the entire summaries.

S: Slightly curved around the sides like up to the main display as well. It was voice activated . **H:** The remote will be single-curved with a cherry design on top. A sample sensor was included to add speech recognition.
S: The market trends and our traditional usual market research study suggests the use of rechargeable batteries. **H:** He suggested substituting a kinetic battery for the rechargeable batteries and using a combination of rubber and plastic for the materials.
S: And in this detailed design the usability interface meeting we will discuss our final design the look-and-feel. **H:** All these components were re-arranged in a revised prototype.

matical, does not tell us anything about what is being spoken about. To obtain a coarse estimate of grammaticality, we also calculate the average log-likelihood score provided by the Stanford Parser. We compute the average log-likelihood scores of the confidence of the dependency parses for each type of summaries. Table 7 shows the average scores.[5]

Table 8 shows some examples of summaries generated by our system. The table indicates that the summaries generated by our system are relatively well-formed and they reflect the formal style of non-conversational text. We aligned sentences from the human-written abstracts and the corresponding sentences selected from each segment that our algorithm generates.

Error Analysis: In some cases the linearization component does not produce relevant ordering of words. For example, in the third summary sentence in Table 8, the system-generated summary lacks certain conjunctions and the ordering of words is inappropriate. The phrase "detailed design the" can be removed to maintain clarity. In addition, the entity – *design* has been repeated. To solve these problems, we plan to introduce intra-sentence level constraints to improve generated summaries.

Our algorithm is designed to retain informative words as well as grammatical dependencies that are more probable in any given corpus. However, the grammatical dependencies that we choose might not necessarily lead to well-formed grammatical sentences. Furthermore, our ILP-based model is unable to understand long-term dependencies of entities within a generated sentence. For example, consider the following output:

"Decided important reflect our budget our the product accessible a wide range of consumers limiting anyone know that kind ."

As can be seen, it is really hard for a reader to identify what the summary sentence is trying to convey although certain words or phrases hint at the topic of *deciding the budget based on the range of consumers*. Our model at present does not memorize previous choices of entities referred in the utterances. Furthermore, our linearization component is based on the ordering of words in the source utterances. However, using the same ordering as the source sentences might not necessarily work well. In the context of an entirely new generated summary sentence, lexical or phrasal reordering and other transformations might be required. In such cases, it might be more effective to use language model based confidence scores to determine the best ordering of words. Improvement in the ILP formulation is possible by including confidence of the sequence of words in addition to the incorporation of knowledge about the entities. Optimizing such a complete model can help generate summaries that are much easier to read. Further, such summaries would contain coherent elements on the same entities in the summary sentences. We might also hope to optimize the model by including the number of segments as a parameter in the model. Currently, the maximum number of sentences in the summary is dependent on the number of topic segments. We can improve the formulation such that model itself decides the optimal length of the summaries ensuring that all the informative points in the meeting discussion are included in the system generated summary.

[5]Lower the magnitude of the log-likelihood scores, the higher is the confidence associated with the dependency parse.

5. CONCLUSIONS AND FUTURE WORK

In this work, we have proposed an approach to generate abstractive summaries from meeting conversations. We proposed a method for dividing a conversation into multiple topic segments. We used an extractive summarizer to identify the important set of utterances, and then applied ILP-based utterance fusion to generate one sentence summary from every topic segment. We leveraged the grammatical relations in the fusion technique that are more dominant in non-conversational style of text. The experiments on content selection and readability indicate that our method can generate relevant abstractive summaries from meeting transcripts without any templates. However, as we have already pointed out, not all generated summaries are usable due to the lack of coherence among several entities discussed within the same summary sentence. We plan to improve the generation using knowledge of entities and also refine readability using a language model. In future work, we plan to develop better linearization techniques. We also plan to improve our algorithm by not limiting one sentence per segment but allowing the ILP model to decide the optimal number of sentences for a complete summary.

6. REFERENCES

[1] C. Apté, F. Damerau, and S. M. Weiss. Automated Learning of Decision Rules for Text Categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251, 1994.

[2] S. Banerjee, P. Mitra, and K. Sugiyama. Abstractive Meeting Summarization Using Dependency Graph Fusion. In *Proc. of the 24th International Conference on World Wide Web Companion (WWW '15 Companion)*, pages 5–6, 2015.

[3] R. Barzilay and K. R. McKeown. Sentence Fusion for Multidocument News Summarization. *Computational Linguistics*, 31(3):297–328, 2005.

[4] B. K. Boguraev and M. S. Neff. Discourse Segmentation in Aid of Document Summarization. In *Proc. of the 33rd Annual Hawaii International Conference on System Sciences (HICSS-33)*, pages 1–10, 2000.

[5] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, et al. The AMI Meeting Corpus: A Pre-announcement. In *Proc. of the 2nd International Workshop on Machine Learning for Multimodal Interaction (MLMI 2005)*, pages 28–39, 2006.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling

Technique. *Journal of Artificial Intelligence Research (JAIR)*, 16(2002):321–357, 2002.

[7] J. Clarke and M. Lapata. Global Inference for Sentence Compression: An Integer Linear Programming Approach. *Journal of Artificial Intelligence Research (JAIR)*, 31(2008):399–429, 2008.

[8] J. Eisenstein and R. Barzilay. Bayesian Unsupervised Topic Segmentation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 334–343, 2008.

[9] K. Filippova. Multi-sentence Compression: Finding Shortest Paths in Word Graphs. In *Proc. of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 322–330, 2010.

[10] K. Filippova and M. Strube. Sentence Fusion via Dependency Graph Compression. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 177–185, 2008.

[11] K. Filippova and M. Strube. Tree Linearization in English: Improving Language Model Based Approaches. In *Proc. of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2009)*, pages 225–228, 2009.

[12] M. Galley. A Skip-Chain Conditional Random Field for Ranking Meeting Utterances by Importance. In *Proc. of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 364–372, 2006.

[13] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. Discourse Segmentation of Multi-party Conversation. In *Proc. of the 41st Annual Meeting on Association for Computational Linguistics (ACL '03)*, pages 562–569, 2003.

[14] N. Garg, B. Favre, K. Riedhammer, and D. Hakkani-Tür. ClusterRank: A Graph Based Method for Meeting Summarization. In *Proc. of the 10th Annual Conference of the International Speech Communication (INTERSPEECH 2009)*, pages 1499–1502, 2009.

[15] D. Gillick, K. Riedhammer, B. Favre, and D. Hakkani-Tur. A Global Optimization Framework for Meeting Summarization. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2009)*, pages 4769–4772, 2009.

[16] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual, 2014.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[18] C. Hori and S. Furui. A New Approach to Automatic Speech Summarization. *IEEE Transactions on Multimedia*, 5(3):368–378, 2003.

[19] P.-Y. Hsueh and J. D. Moore. Automatic Topic Segmentation and Labeling in Multiparty Dialogue. In *Spoken Language Technology Workshop*, pages 98–101, 2006.

[20] C.-Y. Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proc. of the ACL-04 Workshop on Text Summarization Branches Out*, pages 74–81, 2004.

[21] C.-Y. Lin and E. Hovy. The Automated Acquisition of Topic Signatures for Text Summarization. In *Proc. of the 18th Conference on Computational Linguistics (COLING '00)*, pages 495–501, 2000.

[22] H. Lin, J. Bilmes, and S. Xie. Graph-based Submodular Selection for Extractive Summarization. In *Proc. of the IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU 2009)*, pages 381–386, 2009.

[23] F. Liu and Y. Liu. From Extractive to Abstractive Meeting Summaries: Can It Be Done by Sentence Compression? In *Proc. of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP 2009)*, pages 261–264, 2009.

[24] Y. Liu, S. Xie, and F. Liu. Using N-best Recognition Output for Extractive Summarization and Keyword Extraction in Meeting Speech. In *Proc. of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2010)*, pages 5310–5313, 2010.

[25] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014): System Demonstrations*, pages 55–60, 2014.

[26] Y. Mehdad, G. Carenini, F. W. Tompa, and R. T. NG. Abstractive Meeting Summarization with Entailment and Fusion. In *Proc. of the 14th European Workshop on Natural Language Generation*, pages 136–146, 2013.

[27] J. Morris and G. Hirst. Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text. *Computational Linguistics*, 17(1):21–48, 1991.

[28] G. Murray and G. Carenini. Summarizing Spoken and Written Conversations. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 773–782, 2008.

[29] G. Murray, G. Carenini, and R. Ng. Generating and Validating Abstracts of Meeting Conversations: a User Study. In *Proc. of the 6th International Natural Language Generation Conference (INLG 2010)*, pages 105–113, 2010.

[30] G. Murray, G. Carenini, and R. Ng. Interpretation and Transformation for Abstracting Conversations. In *Proc. of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2010)*, pages 894–902, 2010.

[31] A. Nenkova and K. McKeown. A Survey of Text Summarization Techniques. *Mining Text Data*, pages 43–76, 2012.

[32] T. Oya, Y. Mehdad, G. Carenini, and R. Ng. A Template-based Abstractive Meeting Summarization: Leveraging Summary and Source Text Relationships. In *Proc. of the 8th International Natural Language Generation Conference (INLG 2014)*, pages 45–53, 2014.

[33] T. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1-from Yesterday's News to Tomorrow's Language Resources. In *Proc. of the 3rd International Conference on Language Resources and Evaluation Conference (LREC'02)*, pages 827–832, 2002.

[34] D. Roth and W.-T. Yih. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *Proc. of the 8th Conference on Natural Language Learning (CoNLL-2004)*, pages 1–8, 2004.

[35] S. Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading*, pages 225–227, 1990.

[36] L. Wang and C. Cardie. Domain-Independent Abstract Generation for Focused Meeting Summarization. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1395–1405, 2013.

[37] S. Xie and Y. Liu. Using Corpus and Knowledge-based Similarity Measure in Maximum Marginal Relevance for Meeting Summarization. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, pages 4985–4988, 2008.