Automatic Minuting: A Pipeline Method for Generating Minutes from Multi-Party Meeting Transcripts

Kartik Shinde†, Tirthankar Ghosal‡, Muskaan Singh*, and Ondřej Bojar‡
†Indian Institute of Technology Patna, Bihta, Bihar, India
‡Charles University, Faculty of Mathematics and Physics, ÚFAL, Czech Republic
*Speech and Audio Processing Group, IDIAP Research Institute, Martigny, Switzerland
kartik1901ce16@iitp.ac.in, msingh@idiap.ch
(ghosal,bojar)@ufal.mff.cuni.cz

Abstract

Automatically generating meeting minutes is a challenging yet time-relevant problem in speech and natural language processing. Nowadays, meeting minutes seem more crucial than ever due to the manifold rise of online meetings. However, automatic minuting is not straightforward for various reasons: obtaining transcriptions of sufficient quality, summarizing long dialogue discourse, retaining topical relevance and coverage, handling redundancies and small talk, etc. This paper presents our investigations on a pipelined approach to automatically generate meeting minutes using a BART model (Bidirectional and Auto-Regressive Transformers) trained on multi-party dialogue summarization datasets. We achieve comparable results with our simple yet intuitive method with respect to previous large and computationally heavy state-of-the-art models. We make our code available at https://github.com/ELITR/minuting-pipeline.

1 Introduction

Ever since most of our interactions went virtual, the need for automatic support to run online meetings became essential. Due to frequent meetings and the resulting context switching, people are experiencing an information overload (Fauville et al., 2021) of epic proportions. Hence a tool to automatically summarize a meeting transcript would be a valuable addition to the virtual workplace. Automatic Minuting is the task of generating bullet-point meeting minutes from multi-party meeting transcripts. The AutoMin shared task at Interspeech 2021 (Ghosal et al., 2021) is a community-wide effort in this direction. Organizers of AutoMin (Ghosal et al., 2022a) released a medium-scale annotated corpus (Nedoluzhko et al., 2022) of transcript-minute pairs for conducting the shared task.

Automatic Minuting is close to summarization but not the same; subtle differences exist. Summarization aims at generating a concise and coherent text summary. It often purposely removes some less critical information; minuting is more inclined towards adequately capturing the entire contents of the meeting (coverage is probably more significant than coherence and conciseness).

Summarizing spoken multi-party dialogues comes with challenges: incorrect or noisy automated speech recognition (ASR) outputs, long discourse, topical shifts, the dialogue turns, redundancies and small talk, etc. Hence we deem automatic minuting to be more difficult than text summarization.

Due to the variety of sub-problems associated with this task, we adopt a pipelined approach. Our method encompasses (i) pre-processing the ASR-generated meeting transcripts to drop redundancies and noise, followed by (ii) unsupervised topical segmentation, and finally (iii) summarizing each segment of the discourse with a BART model (Raffel et al., 2019) pre-trained on a large-scale dialogue summarization dataset. Our initial investigation yields encouraging results. The obtained minutes resemble the human gold standard in terms of readability and coverage. Our main contribution lies in developing a lightweight, easy-to-implement, and efficient au-
automatic minuting pipeline by leveraging pre-trained Transformer-based language models fine-tuned on large-scale dialogue summarization datasets.

2 Related Work

Although meeting summarization is a well-studied problem in the summarization literature (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017; Celikyilmaz et al., 2018; Chen and Bansal, 2018; Zhong et al., 2019; Xu and Durrett, 2019; Liu and Lapata, 2019; Lebanoff et al., 2019; Cho et al., 2019; Wang et al., 2020; Xu et al., 2019; Jia et al., 2020), automatic minuting is defined as a task relatively recently (Ghosal et al., 2021). We survey some of the relevant meeting summarization research in this section.

Early studies like Chen and Metze (2012) used intra-speaker topic modeling to summarize meetings. Later, several approaches (Zhao et al., 2019; Liu and Chen, 2019; Liu et al., 2019) documented the efficacy of hierarchical methods in learning the inherent structure of conversations. Li et al. (2019) utilized a multi-modal hierarchical attention mechanism across the topic, utterance, and word levels for the task. However, their method depends on manual annotation of topical segments and visual attention of the participants in the meetings, which are not commonly available. Zhu et al. (2020) introduced a hierarchical network HMNet for end-to-end training with cross-domain flexibility, which is now one of the state-of-the-art models for meeting summarization but is very resource-intensive. Recently, Liu and Chen (2021) proposed a dynamic sliding window strategy for abstractive summarization, achieving a close to state-of-the-art performance. Along similar lines, Zhong et al. (2021) presented a pre-training approach for long dialogue understanding and summarization with window-based denoising. Zhang et al. (2021) introduced a flexible multi-stage framework for longer input texts, combining a multi-stage greedy transcript segmentation with end-to-end training. Singh et al. (2021) tested several baseline text summarization models for automatic minuting and concluded that off-the-shelf summarization models are not suited for the concerned task.

Most of the above deep neural models are resource-heavy. The hierarchical model, HMNet, requires 4 Tesla V-100 GPUs with 32G memory on each. Our proposed pipeline approach is straightforward and consists of separate stages for each sub-task in the pipeline: pre-processing, redundancy elimination, transcript segmentation, summarization, and post-processing. Each stage has a unique problem, with specified target outputs, culminating in the final objective, i.e., minutes generation. We would also like to point out that the earlier methods do not aim for automatic meeting minutes generation; instead, they strive to generate coherent meeting summaries in the form of paragraphs. Our motivation is to generate meeting minutes in the form of bullet points that adequately capture the contents of the meeting.

3 Methodology

Our current approach is inspired by one of the system submissions (Shinde et al., 2021) in the AutoMin shared task (Ghosal et al., 2021). Initially, we pre-process the transcripts as described in Section 3.1, later utilize the fine-tuned dialogue summarization model (Section 3.2), and finally, we post-process the outputs (Section 3.3). We describe the datasets used for the fine-tuning and evaluation in Section 4. We also provide automatic and human evaluation discussions and error analysis in Section 5. Kindly refer to Figure 1 for the entire system architecture.

3.1 Pre-Processing

Raw transcripts (directly from the ASRs) would require a good amount of pre-processing before one can proceed with the downstream tasks (automatic minuting in our case). In our experiments, the raw transcripts were already processed by human annotators to remove any inconsistencies during the respective corpora development. We discuss the steps employed for our use case on the already processed datasets.

Redundancy Elimination. Since current summarization models are not trained to eliminate redundancies and are often capped to specific input lengths, they struggle to process a long sequence of multi-speaker utterances and the dispersed information that comes with them (Ghosal et al., 2022b). We leverage specific pre-processing methods and em-
So, in addition to this issue about the UW stuff there was announced today, via the LDC, a corpus from I believe Santa Barbara.

I saw it. I've been watching for that corpus of general spoken English.

...
ploy utterance cleaning and redundancy elimination based on thresholds to tackle this issue.

Consider a transcript with speaker-utterance pairs, \( X^0 = \{(p_1^0, U_1^0), (p_2^0, U_2^0), \ldots, (p_L^0, U_L^0)\} \), where \( p_j^0 \subset P \), \( 1 \leq j \leq L \), is the j-th speaker and \( U_j^0 = (w_1^0, w_2^0, \ldots, w_{L_j}^0) \) is the tokenized sequence of the j-th utterance; where \( \{w_i^j\} \) represents the i-th token from the j-th utterance. For the j-th tokenized utterance, \( \{w_i^j\} \) is the tokenized sequence of the j-th speaker and \( U_j^0 = (w_1^0, w_2^0, \ldots, w_{L_j}^0) \) from the transcript, we generate a cleaned sequence, \( U_j^c = (W_1^j, W_2^j, \ldots, W_{L_j}^j) \), by eliminating repetitions, pauses and known special symbols for unarticulated sounds, unintelligibility, disfluency markers, and similar disruptions. We filter the utterances using custom stopwords set \( S \) that we define from various meeting transcripts from currently available corpora like AMI (McCowan et al., 2005), ICSI (Janin et al., 2003), and the dataset from AutoMin (Nedoluzhko et al., 2022). By this, we obtain the filtered utterance \( U_j^f = (U^c, S) \) and the corresponding context ratio \( R \), which expresses how much the utterance was shortened by dropping stopwords compared to the cleaned version:

\[
R = \frac{|U_j^f|}{|U^c|}
\]

Ultimately, our processed transcript \( X' \) comprises utterances \( U_i^c \) where the ratio of non-stop-words \( R_i \) is big enough, i.e. \( R_i \geq \alpha \) (\( \alpha \) being a predefined threshold ratio).

**Linear Segmentation.** Current summarization models limit the length of input sequences they can process (Singh et al., 2021), so they cannot process the full-length transcripts in our data. Our approach here is simple: it breaks the transcripts into blocks with a uniform token length. We experiment with token lengths: 512, 768, and 1024, respectively.

**Topical Segmentation.** The linear segmentation technique is problematic whenever important information on a topic falls into the subsequent segment. To address this limitation, we experiment with two methods for topic-aware segmentation: Depth-Scoring (adopted from Solbiati et al. (2021)) and the TextTiling algorithm by (Hearst, 1993).

For Depth Scoring, we use a window of \( k_w \) segments, capping each segment to \( L = 60 \) words and setting topic change threshold \( \tau \) to 0.5 (these are tunable hyperparameters, kindly refer to Solbiati et al. (2021) for details). Let us consider Figure 2. For a transcript with \( N \) turns, we obtain their contextualized embeddings from an encoder. We apply max pooling on this embedding space.

For a pair of neighboring windows of segments, one consisting of turns \( k - k_w \) till \( k \) and the other of turns \( k \) till \( k + k_w \), we obtain the cosine similarity, \( sim_k \) between the embeddings pooled across all segments in the respective windows. For a series of neighbouring window similarity scores \( s = (sim_{k_w}, \ldots, sim_{N-k_w}) \), we compute the depth scores as \( dp_k = \frac{hl(k) + hr(k) - 2sim_k}{\tau} \) where \( hl(k) \) and \( hr(k) \) are the highest similarity score on the left and right side of the \( k \)th element in the series of similarity scores. We deduce the topic change indices with the help of the obtained window-similarity scores and depth scores. Following are the variations one can use while determining the topic change indices.

- **Segment-window capping.** With this approach, we compute the topic change indices as:

\[
T_{ds} = \{i \in [0, M] | sim_{k_w+i} \leq \mu_s - \sigma_s \}
\]

where \( T \) is the set of topic-change indices \( \mu_s \) and \( \sigma_s \) are the mean and variance of the sequence, \( M = N - k_w \) is the number of windows, \( sim_{k_w+i} \) is the similarity score of the \( i \)th window.

- **TextTiling.** TextTiling is a method to subdivide texts into multi-paragraph units representing passages or subtopics by leveraging lexical co-occurrence and distribution patterns. Here, we use TextTiling to identify major subtopic shifts. After computing the window similarity scores, we use the TextTiling method to compute the segments in a transcript. For a series of depth scores \( D = (d_1, d_2, \ldots, d_{N-k_w}) \), we compute the topic change indices as:

\[
T_{T} = \{i \in [1, M] | d_i \geq \tau \}
\]

Through one of the three approaches (linear, depth-scoring, or text tiling), we obtain the segmented transcript \( X^S = (\eta_1, \eta_2, \ldots, \eta_K) \) where \( \eta_k = \{(p_1^{S_k}, U_1^{S_k}), (p_2^{S_k}, U_2^{S_k}), \ldots, (p_{L_k}^{S_k}, U_{L_k}^{S_k})\} \) is the sequence of speaker-utterance pairs belonging to that segment.
We concatenate the speaker labels $p^i$ with the corresponding utterances $U^i$ and then across all items in the given segment back to the form of a single dialogue transcript. We then pass each of these plain text segments to one of the root summarization modules (see Section 3.2).

### 3.2 Summarization

We choose the pre-trained BART model \cite{Lewis2019BARTAD} in the summarization module in our pipeline. BART performs best among the other summarization models we tested, generating fluent and readable meeting minutes. Other summarization models include T5 \cite{Raffel2019T5} , Pegasus \cite{Zhang2020Pegasus} , and RoBERTa2RoBERTa \cite{Rothe2020RoBERTa2} . We fine-tune all these models on popular dialogue summarization datasets before integrating them into our pipeline.

BART is a denoising autoencoder for pretraining sequence-to-sequence models. The model is trained by corrupting text in an arbitrary noising function and then teaching it to reconstruct the original text. BART’s ability to use source-side bi-directionality when operating on sequence generation tasks encourages its use for text summarization.

We pass the input sequence obtained from the preprocessing module through the summarization module. Again, for k-th segment, it returns a summary $C_k = \{c^k_1, c^k_2, \ldots c^k_l\}$, where $c^k_i$ is the i-th summary line of the k-th segment. We rejoin all the segment summaries $Y^S = (C_1, C_2, \ldots C_K)$ to get the raw summary text.

**Experimental Configuration** We do not train any models from scratch but finetune most of them on
Table 1: Statistics of the dialogue and meeting summarization datasets we employ in our experiments. The top part is the larger summarization datasets we use for fine-tuning our models, the bottom part is the meeting summarization datasets we use for model selection and testing. The reported statistics are averages across entire corpora. Lengths are in words. The compression ratio indicates how much the dialogue is shortened into the summary.

<table>
<thead>
<tr>
<th>Datasets</th>
<th># Dialogues</th>
<th># Turns</th>
<th># Speakers</th>
<th># Turn Len.</th>
<th># Len. of Dialogue</th>
<th># Summary Len.</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMSum</td>
<td>16.4K</td>
<td>11.2</td>
<td>2.4</td>
<td>9.1</td>
<td>124.0</td>
<td>23.4</td>
<td>81.12%</td>
</tr>
<tr>
<td>DialogSum</td>
<td>13.5K</td>
<td>9.5</td>
<td>2.0</td>
<td>15.8</td>
<td>168.5</td>
<td>25.8</td>
<td>84.70%</td>
</tr>
<tr>
<td>MediaSum</td>
<td>463.6K</td>
<td>30.0</td>
<td>6.5</td>
<td>49.6</td>
<td>1553.7</td>
<td>14.4</td>
<td>99.00%</td>
</tr>
<tr>
<td>AMI</td>
<td>137</td>
<td>535.6</td>
<td>4.0</td>
<td>10.4</td>
<td>5570.4</td>
<td>321</td>
<td>94.24%</td>
</tr>
<tr>
<td>ICSI</td>
<td>59</td>
<td>819.0</td>
<td>6.3</td>
<td>10.5</td>
<td>8567.7</td>
<td>576</td>
<td>93.28%</td>
</tr>
<tr>
<td>ELITR Corpus</td>
<td>124</td>
<td>254.4</td>
<td>5.8</td>
<td>9.7</td>
<td>8890.8</td>
<td>387</td>
<td>95.65%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Instances</th>
<th>Doc. Len.</th>
<th>Summ. Len.</th>
<th>% Comp.</th>
<th>% novel unigram</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSum</td>
<td>226.0K</td>
<td>488</td>
<td>27</td>
<td>94.5%</td>
<td>37.8%</td>
</tr>
<tr>
<td>CNN/DM</td>
<td>311.0K</td>
<td>906</td>
<td>63</td>
<td>93.0%</td>
<td>16.9%</td>
</tr>
<tr>
<td>R-TIFU</td>
<td>7.9K</td>
<td>641</td>
<td>65</td>
<td>89.9%</td>
<td>43.8%</td>
</tr>
</tbody>
</table>

Table 2: Document summarization datasets used for fine-tuning.

the data described in Section 4 below. For most models, a single Tesla K80 GPU is sufficient. Few larger models like BART-large and T5-large require multi-GPU training on NVIDIA GTX 1050 Ti or single GPU training on the NVIDIA A100-PCI-E-40GB variant. Training for individual finetuning procedures takes less than 2 hours, while warm-starting takes approximately 0.5 hours, depending on the dataset used. The hyperparameters and model configurations are consistent with the default values used during the pretraining of respective models. We set the finetuned BART on inference and generate our text with num_beams = 4, top_k = 0.5 and no limit on ‘max_length’. We provide the hyperparameters and model configuration details in our code repository.

3.3 Post-Processing

After the main summarization, we use sentence compression methods, including swapping shortened phrases and pronouns and splitting longer sentences into two for improved readability. In our proposed pipeline, for each summary line, we filter out a set of unique entities (speaker names, project/corporation names, and location details). Further, we use a token-count threshold \( \tau_{\text{token}} \) of 10 to include only those summary-sentences which are quantitatively informative enough (i.e., consisting of a minimum of \( \tau_{\text{token}} \) number of tokens).

4 Dataset Description

Our work uses two types of data sources (see Table 1): (1) for fine-tuning summarization models, see Section 4.1, and (2) for the choice of the best setup and final evaluation of the minuting task, see Section 4.2.

4.1 Datasets for Fine-tuning Summarization Module

Here, we choose from some of the popular abstractive summarization datasets. Primarily, we use the dialogue summarization corpora: SAMSum (Gliwa et al., 2019), DialogSum (Chen et al., 2021), and MediaSum (Zhu et al., 2021).

Additionally, we use document summarization datasets XSum (Narayan et al., 2018), CNN/DM (Nallapati et al., 2016) and R-TIFU (Kim et al., 2018), see Table 2. Their high compression ratio (“% Comp.”) can potentially train the models to generate sequences more selectively, thus automatically eliminating redundancies.

4.2 Target Datasets: Automatic Minuting/Meeting Summarization

We primarily use ELITR Minuting Corpus (Nedoluzhko et al., 2022) for comparison with other systems. We further experiment on popular meeting summarization datasets: AMI (McCowan et al., 2005) and ICSI (Janin et al., 2003), due to similarities in the two tasks. AMI and ICSI come from staged product design meetings in companies, academic group meetings in schools, and similar arrangements. Each instance has a transcription of the entire dialogue and is annotated with a meeting summary and human-identified topic boundaries (except for ELITR Minuting Corpus). These meeting transcripts are extremely long, have a
Table 3: ROUGE-1, ROUGE-2, ROUGE-SU4 scores of generated summaries on AMI and ICSI datasets. *→ ‘bart-xsum-samsum’ stands for our proposed model finetuned on the XSum corpus, further finetuned on the SAMSum corpus. Results in (C) are reproduced from the respective papers.

<table>
<thead>
<tr>
<th>Models/Metrics</th>
<th>AMI</th>
<th></th>
<th>ICSI</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) Baselines and Comparing Systems</td>
<td>R-1 R-2 R-SU4</td>
<td>R-1 R-2 R-SU4</td>
<td>R-1 R-2 R-SU4</td>
<td>R-1 R-2 R-SU4</td>
</tr>
<tr>
<td>Random</td>
<td>35.13 6.26 13.17</td>
<td>35.14 6.46 13.35</td>
<td>29.28 3.78 10.29</td>
<td>27.64 3.68 9.77</td>
</tr>
<tr>
<td>Cluster Rank (Garg et al., 2009)</td>
<td>Extractive Oracle</td>
<td>39.49 9.65 13.20</td>
<td>34.66 8.00 10.49</td>
<td>32.00 7.70 14.46</td>
</tr>
<tr>
<td>PGNet (See et al., 2017)</td>
<td>(B) Our best-performing setups</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bert2bert-cnndm-samsum</td>
<td>40.72 10.10 27.13</td>
<td>35.03 7.35 24.48</td>
<td>45.17 13.30 20.33</td>
<td>38.75 8.51 14.98</td>
</tr>
<tr>
<td>bart-xsum-dialogsum</td>
<td>42.40 10.34 17.67</td>
<td>36.95 6.94 13.68</td>
<td>32.00 7.70 14.46</td>
<td>42.71 11.05 18.34</td>
</tr>
<tr>
<td>t5-dialogsum</td>
<td>42.71 11.05 18.34</td>
<td>37.01 7.48 13.68</td>
<td>32.00 7.70 14.46</td>
<td>24.48</td>
</tr>
<tr>
<td>bart-xsum-samsum*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(C) State-of-the-art systems in Meeting Summarization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMNet (Zhu et al., 2020)</td>
<td>53.02 18.57 24.85</td>
<td>46.28 10.60 19.12</td>
<td>37.01 7.48 13.68</td>
<td>32.00 7.70 14.46</td>
</tr>
<tr>
<td>DialogLM (Zhong et al., 2021)</td>
<td>53.70 19.60 -</td>
<td>49.50 12.50 -</td>
<td>37.01 7.48 13.68</td>
<td>37.01 7.48 13.68</td>
</tr>
<tr>
<td>Summ’ (Zhang et al., 2021)</td>
<td>53.40 20.30 -</td>
<td>48.80 12.20 -</td>
<td>37.01 7.48 13.68</td>
<td>37.01 7.48 13.68</td>
</tr>
</tbody>
</table>

Table 1 shows the relevant statistics of the dialogue and meeting summarization datasets that we use in our experiments.

5 Evaluation

In this section, we present the evaluation of our proposed pipeline in terms of automatic metrics in Section 5.1 and human evaluation metrics in Section 5.2. We compare our proposed pipeline with different summarization algorithms, finetune on combinations of abstractive summarization datasets, and report our performance on ELITR Minuting Corpus, AMI, and ICSI meeting summarization datasets.

5.1 Automatic Evaluation

For automatic evaluation, we make use of popular text summarization evaluation metrics. We report ROUGE (Lin, 2004) variants, namely ROUGE-1, ROUGE-2, ROUGE-SU4, which measures the overlap of unigrams, bigrams, and unigrams plus skip-bigrams (with max. skip of 4), respectively. We also provide METEOR (Banerjee and Lavie, 2005) scores which reward matching stems, synonyms, and paraphrases and not just exact matches.

5.2 Human Evaluation

To evaluate the quality of our output, we carry out a human evaluation of our minutes and compare it with the best-performing model outputs from the AutoMin 2021 shared task. Since we were the AutoMin shared task organizers, we had access to the human evaluators who also evaluated the system submissions in AutoMin. Six human evaluators rated our minutes in terms of Adequacy, Grammaticality and Fluency scores on a Likert scale of 5 (we report the average scores) (Ghosal et al., 2021). Because automatic metrics for text summarization evaluation have various shortcomings and are not apt to judge the quality of meeting minutes (Ghosal et al., 2022b), we attribute more importance to human evaluation, although the annotators were judging only our outputs in this run, without immediate comparison to AutoMin system outputs.

5.3 Results and Analysis

We discuss the experimental results and analyze the performance of our system in this section.

Table 3 compares the ROUGE scores of earlier models with our best setup (bart-XSum-samsum with linear segmentation). With no prior fine-tuning on AMI and ICSI meeting datasets, our pipeline outperforms several earlier approaches, including the popular Pointer Generator network (See et al., 2017) and the Extractive Oracle. However, the state-of-the-art models: HMNet (Zhu et al., 2020), DialogLM (Zhong et al., 2022) and SUMM-N (Zhang et al., 2021) are still superior in terms of the quantitative metrics.

Table 4 compares the automatic and human evalu-
Table 4: Performance of our pipeline in comparison to the two best-performing participating systems at the AutoMin Shared Task on the newly released ELITR Minuting Corpus.

<table>
<thead>
<tr>
<th>Model</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
<th>Adequacy</th>
<th>Grammatical</th>
<th>Fluency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours-bart-xsum-samsum (Current Model)</td>
<td>0.40±0.09</td>
<td>0.11±0.02</td>
<td>0.18±0.03</td>
<td>4.46/5.00</td>
<td>4.45/5.00</td>
<td>4.18/5.00</td>
</tr>
<tr>
<td>Team ABC (Shinde et al., 2021)</td>
<td>0.33±0.08</td>
<td>0.08±0.04</td>
<td>0.19±0.06</td>
<td>3.98±0.73</td>
<td>4.45±0.57</td>
<td>4.27±0.55</td>
</tr>
<tr>
<td>Team Hitachi (Yamaguchi et al., 2021)</td>
<td>0.26±0.09</td>
<td>0.08±0.03</td>
<td>0.14±0.05</td>
<td>4.25±0.46</td>
<td>4.34±0.41</td>
<td>3.93±0.57</td>
</tr>
</tbody>
</table>

Table 5: Comparison of the BART-based model setups with different summarization models on the AMI test set.

<table>
<thead>
<tr>
<th>Model</th>
<th>R-1</th>
<th>R-2</th>
<th>R-SU4</th>
<th>BERTScore</th>
<th>METEOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>bart-xsum-samsum</td>
<td>45.2</td>
<td>13.3</td>
<td>20.3</td>
<td>0.60</td>
<td>20.6</td>
</tr>
<tr>
<td>bart-xsum-dialogsum</td>
<td>42.4</td>
<td>10.3</td>
<td>17.7</td>
<td>0.59</td>
<td>18.6</td>
</tr>
<tr>
<td>bart-base-samsum</td>
<td>39.9</td>
<td>11.2</td>
<td>16.1</td>
<td>0.60</td>
<td>15.1</td>
</tr>
<tr>
<td>bart-base-mediasum</td>
<td>33.2</td>
<td>7.0</td>
<td>11.3</td>
<td>0.55</td>
<td>14.0</td>
</tr>
</tbody>
</table>

Table 6: Comparison of the ‘bart-xsum-samsum’ model with different segmentation methods on AMI dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pk</th>
<th>WinDiff</th>
<th>ROUGE-1</th>
<th>METEOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
<td></td>
<td>0.61</td>
<td>0.75</td>
</tr>
<tr>
<td>TextTiling</td>
<td></td>
<td></td>
<td>0.39</td>
<td>0.41</td>
</tr>
<tr>
<td>Capped</td>
<td>0.34</td>
<td>0.35</td>
<td>42.50</td>
<td>16.7</td>
</tr>
<tr>
<td>Linear (768)</td>
<td>0.44</td>
<td>0.50</td>
<td>45.17</td>
<td>20.6</td>
</tr>
</tbody>
</table>

Table 5 shows the performance of our pipeline when used with different summarization models based on BART on the AMI test set. Our best-performing combination outscores the next by almost 3 points in terms of ROUGE-1; however, other model variants still perform close to the proposed approach. From the setups we tested, the best fine-tuning procedure starts with XSum and continues with the SAMSum dataset.

As we mentioned earlier, our model fine-tuned on the SAMSum corpus offers a better generation quality than those trained on other datasets. We attribute this to the fact that the dialogues in the SAMSum dataset are relatively simplistic and much more straightforward than those in DialogSum and MediaSum. The conversations are comparatively shorter and better reflect a conventional multiparty dialogue situation, leading to a better match between the training and testing conditions.

We also notice the differences caused by the training datasets used before the finetuning phase. Having a high compression ratio and novel word percentage, datasets like XSum demonstrate an extremely abstractive nature of summarization. Although the source text in XSum (Narayan et al., 2018) is longer than the dialogue instances from datasets like SAMSum (Gliwa et al., 2019), the summaries are relatively shorter. We observe a similar difference when we train the model on the XSum dataset compared to other datasets like the Reddit-TIFU (Kim et al., 2018) and the CNN/DailyMail (Nallapati et al., 2016). The generated minutes are relatively short, with more novel words and paraphrased sentences, qualities that are critical in abstractive summarization.

Furthermore, we carry out ablation experiments to check the effectiveness of our three segmentation methods (Table 6). We report the performance on the AMI dataset, which comes with reference segmentation of transcripts. We use the Pk (Befferman et al., 1999) and WinDiff (Pevzner and Hearst, 2002) methods to evaluate the segmentation accuracy and report ROUGE-1 and METEOR scores on AMI transcripts.

We find ROUGE (Lin, 2004) and METEOR (Banerjee and Lavie, 2005) score surprisingly higher with linear segmentation (segments of at most 768 tokens). This result can be due to: with the topical segmentation schemes, several snippets of small talk and irrelevant information from the meeting make it to the final output, thereby affecting the overall scores.

1Ground truth segmentation of meeting transcripts are available for the AMI corpus.
Figure 3: Sample output from our pipeline on a document from AMI corpus: Meeting Id-ES2014b

5.4 Error Analysis
We qualitatively examine and find that our outputs show the following categories of errors (Figure 4).

- **Made-up entities.** Anonymization of discrete entities in transcripts (e.g., LOCATION7, PERSON4, Marketing Manager) is consistent in most transcripts and minutes of our test datasets. Since no such anonymization is apparent in SAMSum, this sometimes results in the generation of made-up entities that are initially not part of that transcript.

- **Absence of context in summary.** Sometimes, the generated summary could use pronouns or other referring expressions from the transcript without ensuring that the element they are referring to is actually present in the summary. However, this issue is rare and did not occur in our final test runs.

- **Incomplete phrases.** Although less, we notice occurrences of incomplete sentences. These generally belong to those parts of the transcripts where the utterances either had missing punctuation or hesitations and interruptions on the speaker’s part.

6 Conclusion
In this paper, we explore the use of large pre-trained language models fine-tuned on dialogue summarization datasets to automatically generate meeting minutes. We evaluate our proposed BART-based pipeline approach on the recently released corpus for automatic minuting (ELITR Minuting Corpus) as well as on the earlier AMI and ICSI meeting summarization corpora. We utilize existing multiparty meeting summarization datasets.

Our pipelined approach is promising and certainly puts up a case for further investigations to employ large language models for this challenging task. In future work, we would like to optimize our existing pipeline by replacing extractive filtering and utterance-level topic segmentation with an end-to-end method.

Acknowledgement
This work has received funding from the grant 19-26934X (NEUREM3) of the Czech Science Foundation.
Figure 4: Error instances from the pipeline-generated summaries illustrating the error cases discussed in Section 5.4.


