Action-Item-Driven Summarization of Long Meeting Transcripts

Logan Golia
Rice University
USA
lsg3@rice.edu

Jugal Kalita
University of Colorado, Colorado Springs
USA
jkalita@uccs.edu

ABSTRACT

The increased prevalence of online meetings has significantly enhanced the practicality of a model that can automatically generate the summary of a given meeting. This paper introduces a novel and effective approach to automate the generation of meeting summaries. Current approaches to this problem generate general and basic summaries, considering the meeting simply as a long dialogue. However, our novel algorithms can generate abstractive meeting summaries that are driven by the action items contained in the meeting transcript. This is done by recursively generating summaries and employing our action-item extraction algorithm for each section of the meeting in parallel. All of these sectional summaries are then combined and summarized together to create a coherent and action-item-driven summary. In addition, this paper introduces three novel methods for dividing up long transcripts into topic-based sections to improve the time efficiency of our algorithm, as well as to resolve the issue of large language models (LLMs) forgetting long-term dependencies. Our pipeline achieved a BERTScore of 64.98 across the AMI corpus, which is an approximately 4.98% increase from the current state-of-the-art result produced by a fine-tuned BART (Bidirectional and Auto-Regressive Transformers) model.[1]

CCS CONCEPTS

• **Information systems** → **Summarization**; • **Computing methodologies** → **Natural language processing**; *Natural language generation*; *Information extraction.*

KEYWORDS

neural networks, text summarization, topic segmentation, action item extraction

ACM Reference Format:
Logan Golia and Jugal Kalita. 2023. Action-Item-Driven Summarization of Long Meeting Transcripts. In *2023 7th International Conference on Natural Language Processing and Information Retrieval (NLPIR 2023), Dec. 15–17, 2023, Seoul, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3639233.3639253

[1] https://github.com/logangolia/meeting-summarization

NLPIR 2023, Dec. 15–17, 2023, Seoul, Republic of Korea

ACM ISBN 979-8-4007-0922-7/23/12…$15.00
https://doi.org/10.1145/3639233.3639253

1 INTRODUCTION

As a result of the COVID-19 pandemic, many professional meetings and conversations have been conducted online; this also means that the transcripts of these meeting have become readily available. Minutes are accepted official records of what transpired in a meeting, and so designated personnel usually conduct the tedious process of generating meeting minutes. However, with the help of large language models (LLMs), we can automate this process and still generate factual and informative summaries.

There are two main approaches to text summarization: extractive and abstractive. Extractive summarization techniques locate the most important phrases and sentences from the input transcript and concatenate them to form a concise summary. However, the summaries generated by these techniques are usually awkward to read because of the forceful concatenation of unrelated sentences [11]. Abstractive summarization techniques focus more on understanding the overall meaning of a transcript and then generating a concise summary based on the entire text. Unlike extractive summarization, abstractive summarization generates new words and phrases that were not found in the input transcript, rather than simply extracting the important phrases [18]. Abstractive summarization is more challenging, but as expected, it leads to better summaries [9]. As a result, meeting summarization has begun to head in this direction, and this study utilizes abstractive summarization techniques as well.

Current approaches to automating the creation of meeting minutes treat summarizing a meeting the same way they would summarize a dialogue [7]. However, we argue that meeting summarization is fundamentally different from dialogue summarization. Unlike a dialogue, useful meeting minutes have additional features that are often not included in the automated summary of the meeting: action items, main topics, tension levels, decisions made, etc. In this study, we focus on incorporating action items into the machine-generated summaries.

LLMs today still struggle to capture long-term dependencies in texts, and as a result, they are not very good at generating summaries for long transcripts [4]. The time and space complexities of these transformer-based models increase quadratically with respect to the input size [23], and new LLMs still have strict input token limits [26]. Most solutions to these problems employ linear segmentation, where the long texts are broken up into equal subsections based on token numbers, but the problem with this approach is that we inevitably interrupt ideas in the text. We build upon previous work in topic segmentation to divide the text into topical chunks before summarizing.

In summary, current solutions to the problem of automatically generating meeting minutes given the transcript of the meeting produce general and vague summaries. In addition, there is a lack

of effective topic segmentation methods for meeting summarization. This study outlines a novel method of utilizing topic segmentation and recursive summarization to generate action-item-driven abstractive summaries of long meeting transcripts.

Our main contributions are threefold:

1) We develop three novel topic segmentation algorithms, in which the best outperforms the summarization performance provided by linear segmentation by 1.36% in terms of the BERTScore metric;

2) We develop our own effective action-item extraction algorithm;

3) Our novel parallel and recursive meeting summarization algorithm properly generates action-item-driven summaries and improves upon the performance of current state-of-the-art models by approximately 4.98% in terms of the BERTScore metric.

2 RELATED WORK

In this section, we discuss previous methods employed in meeting summarization and provide motivation for our novel techniques.

2.1 Recursive Summarization

One way in which meeting summarization differs from dialogue summarization is that meeting transcripts are generally long, and as explained earlier, transformer-based models struggle with larger input sizes. As a result, it has been proven effective to divide long documents into multiple parts, summarize each component, and then combine the summaries back together in a recursive approach. The recursive algorithm described in this paper is inspired by the method proposed by Wu et al. [24], which was used to summarize long books. The methods proposed by Shinde et al. [20] and Yamaguchi et al. [25] are not truly recursive because after they combine the sectional summaries back together, the final summary is never fed back into the summarization model. Instead, they perform argument mining on the resulting chunk of the combined summaries. We propose a truly recursive approach and achieve state-of-the-art results with this technique.

2.2 BART Model for Meeting Summarization

While there do exist more powerful dialogue summarization models such as DialogLM [29] and Summ^N [28], we use the BART (Bidirectional and Auto-Regressive Transformers) model [12] due to its speed and high performance in long document summarization tasks [11]. In addition, there has been previous research in assessing different topic segmentation methods on the BART model, so this allows us to evaluate our techniques.

2.3 AMI Dataset

The AMI dataset is a large meeting corpus consisting of 137 scenario-driven meetings and their corresponding summaries [15]. Even though the scenarios are artificial, the way in which the actors choose how to interact with each other is spontaneous. The realistic meeting conversations combined with the fact that there are 137 different long meeting transcripts makes the AMI corpus an ideal dataset on which to test our techniques on.

2.4 Current Segmentation Techniques

There are several techniques to divide meeting transcripts into multiple parts, but none have actually been able to improve summarization results when compared to the simplest technique, linear segmentation. Linear segmentation is the process of dividing the meeting transcript into parts solely based on token number, including a preset number of tokens in each chunk. The state-of-the-art results on summarizing the AMI corpus using the BART model are achieved through this technique by Shinde et al. [20]. Shinde et al. [20] attempted to use two additional topic segmentation techniques, Depth-Scoring [21] and TextTiling [10], but neither were able to improve upon the results obtained by linear segmentation. Yamaguchi et al. [25] also introduces a novel technique for topic segmentation using a Longformer+LSTM model to predict whether a sentence is the start of a new topic, in the middle of a topic, or outside of a particular topic. However, their summarization results were significantly lower than those achieved by Shinde et al. [20]. We propose three novel segmentation techniques that outperform linear segmentation.

2.5 Evaluation Metrics

ROUGE's F1 scores are the most popular metrics in evaluating machine-generated summaries [13]. However, ROUGE scores have many flaws since they focus solely on the lexical overlap between the machine-generated summaries and the human reference summaries rather than their semantic similarity [5]. As a result, BERTScore, which measures the semantic similarity between the machine-generated summaries and the reference summaries has been growing in popularity [18]. We employ the BERTScore metric as well, since it has been shown to achieve higher correlations with human judgment on the quality of a machine-generated summary compared to ROUGE [27].

3 APPROACH

In this section, we dive deeper into our recursive algorithm for generating action-item-driven meeting summaries. We also explore the lower-level techniques that were necessary to improve state-of-the-art results and provide motivation for these design decisions along the way.

3.1 Divide-and-conquer

As described in our "Introduction" and "Related Works" sections, the first step to summarizing long meeting transcripts is to break them up, so we can summarize each chunk. We propose three simple but effective topic segmentation techniques that were able to generate more truthful and concise summaries when compared to linear segmentation.

3.1.1 Chunked Linear Segmentation. When we ran our model using linear segmentation (splitting the text based on a preset token number across all chunks), we noticed that points were often misunderstood and repeated because we were creating separate chunks in the middle of a speaker's formulation of one idea; let us call each speaker's contiguous dialogue a "turn." Thus, we first employed a simple technique inspired by linear segmentation where we maximize the number of tokens in each chunk, adding

turns until we reach a preset token number, whilst ensuring that no speaker's turn is interrupted.

3.1.2 Simple Cosine Segmentation. The second technique we created is based upon chunked linear segmentation, but also upon the cosine similarity of the MPNet embeddings, a state-of-the-art sentence embedding model [22], for each turn. For each turn, we compute its MPNet embedding and calculate its cosine similarity with the MPNet embedding of the previous turn. If the cosine similarity of the embeddings is greater than 0, we simply add this turn to the current chunk. If the cosine similarity of the embeddings is less than or equal to 0, we define the current turn to be the beginning of a new topic and start a new chunk.

We choose a similarity threshold of 0 to signify the start of a new topic after experimenting with different values and manually inspecting the quality of the resulting summaries, as well as evaluating the resulting summaries with the ROUGE and BERTScore metrics. This value of 0 also makes sense in theory because it means that the two consecutive turns are more semantically dissimilar than they are similar. This leads to better results because we do not want to split the transcript into too many topics, and instead favor large topics; we generally want to keep as much text intact as possible, so the summarization model has enough context to generate a quality summary. This is also why topic segmentation for summarization is very different from typical topic segmentation because we do not want to create chunks at every little topic change. In fact, when we increased our similarity threshold from 0 to just 0.2, our BERTScores and ROUGE-L scores both decreased by $> 1\%$ which is very significant for summarization tasks.

It is also important to note that when splitting based on some cosine similarity threshold, there is a risk that no new chunks will be created for over 1024 consecutive tokens, which is the max input token limit for the BART model [17]. Therefore, as we move through the turns and add them to the existing chunk, we check to ensure that adding the current turn will not make the current chunk greater than 1024 tokens. If this does happen, we create a new chunk/topic beginning with this turn, regardless of this turn's cosine similarity with the previous turn.

3.1.3 Complex Cosine Segmentation. We noticed a recurring problem when inspecting the topic chunks that were being created by the previous method. Sometimes a person would utter something meaningless, and that would compose their entire turn (e.g. "Bob: Ummm"). As a result, this turn would often have a very low cosine similarity with the previous turn, and a new topic/chunk would be created. The simplest solution to this problem would be to remove all redundant and meaningless utterances in the pre-processing stage. The problem with this approach is that even if we somehow managed to hard code the regular expressions in order to remove all of the "meaningless" turns, there are still lots of cases where a speaker will say something completely unrelated to the current topic (e.g. "Bob: Let us go grab ice cream after this"), but then they will resume talking about the original topic. In this case, we would not want to create a new topic. In order to achieve this, we take the same approach used in "simple cosine segmentation", except we recalculate the MPNet embedding of the entire current chunk before comparing its cosine similarity to the MPNet embedding of the following turn. This mitigates the effect of "meaningless" turns, particularly consecutive "meaningless" turns, since they will have less impact on the MPNet embedding of the chunk we are comparing the next turn to. Please refer to Algorithm 1 for further details.

3.2 Generating the General Sectional Suammries

Once we have divided the original text into chunks, the next step is to generate a general abstractive summary for each chunk. Our approach to solve this problem involves fine-tuning Meta's BART model [12], a pre-trained large language model, on dialogue datasets to generate general summaries of a meeting. We elect to use a BART model since its bidirectional encoder and auto-regressive decoder have been shown to better understand the full semantics of a text and generate coherent summaries [12]. Specifically, we used a BART model fine-tuned on the XSUM [16] and SAMSUM [8] datasets to generate the general summaries for each chunk. These are widely used dialogue datasets for training dialogue summarization models [6]. They are also the same datasets on which Shinde et al. [20] fine-tuned their model, so we can better compare our results.

In addition, we noticed that since each general sectional summary is independent of one another, they can be generated in parallel. To the best of our knowledge, we are the first to incorporate parallelism in the divide-and-conquer summarization algorithm as seen in Algorithm 3.

3.3 Action-Item Extraction

Another very important component of any good meeting summary is what each participant has accomplished and what they need to accomplish before the next meeting; so for each chunk of text, we need to extract the action items. Although recording action items is an important part of many meeting summaries, the issue has been ignored in prior work. This problem was first introduced by Cohen et al. [2], but little progress has been made since. To solve this, we use a public dataset[2] from a GitHub repository that contains 2750 dialogue statements as well as corresponding labels for whether a statement contains action items or not. We then fine-tune a BertForSequenceClassification[3] model (a BERT model [3] with a linear layer on top for classification) on this dataset to classify the action items in the original meeting transcript. This training method proved effective with a classification accuracy of 95.4% on the test dataset. However, this process alone is not enough to extract the key action items from a text. This method alone identifies which sentences contain action items, but it does not extract the underlying ideas. For example, a sentence identified as an action item can be "you need to do that before the next meeting." This is indeed an action item, but it doesn't actually contain any useful information; there are too many pronouns and not enough context. In the next sections, we discuss existing methods to solve this problem, explain their limitations, and present our own technique.

3.3.1 Coreference Resolution. We first employed widely used state-of-the-art methods and models for coreference resolution to convert the sentences that were classified as action items into more

[2]https://github.com/kiransarv/actionitemdetection/blob/master/dataset
[3]https://huggingface.co/docs/transformers/v4.31.0/en/model_doc/bert#transformers.BertForSequenceCl

Algorithm 1 Complex Cosine Segmentation(string text, int similarityThreshold, int maxTokens)

```
turns ← text split by speaker
model ← sentence embedding model
tokenizer ← tokenizer used by summarization model
processedChunks ← list with the first sentence from turns
for i in range(1, len(turns)) do                          ▷ Iterate through the turns
    curChunkEmbedding ← model.encode(processedChunks[−1])
    nextSpeakerEmbedding ← model.encode(turns[i])
    similarity ← cosineSimilarity(curChunkEmbedding,nextSpeakerEmbedding)   ▷ Compute similarity
    newChunk ← processedChunks[−1] + turns[i]
    newNumTokens ← tokenLen(tokenizer(newChunk))
    if similarity > similarityThreshold and newNumTokens ≤ maxTokens then
        processedChunks[−1] ← newChunk                    ▷ Add turn to the current chunk
    else
        append turns[i] to processedChunks                ▷ Start a new chunk
    end if
end for
return processedChunks                                    ▷ A list of topic-based chunks of text
```

context-rich statements. We employed libraries such as Stanford CoreNLP [1] and NeuralCoref[4] (an extension of the spaCy library), but were not satisfied by the results. Not only were the pronouns not always resolved for larger text inputs, but we realized that coreference resolution alone was not enough. Even if the pronouns were resolved, this was often not enough context to completely understand the sentence containing the action item. For example, the sentence "you need to do that before the next meeting" may be converted to "Jake needs to fix the website before the next meeting" after coreference resolution. This is better, but it is still not enough information for Jake to read this sentence in the meeting minutes and understand what needs to be done.

3.3.2 Context Resolution. In this paper, we develop a technique to solve this lack-of-context problem which we call "neighborhood summarization." Once we find a sentence that has been identified as an action item, we find its "neighborhood." We define a sentence's neighborhood as the three sentences before the sentence, the sentence itself, and the two sentences after the sentence. We use all six of these sentences as inputs into the same BART summarization model that we used to generate the sectional summaries, and we are left with a rephrased version of the sentence containing the action item. We believe the reason this technique works so well is because the reference summaries in the dialogue datasets that our BART model is fine-tuned on are naturally action-item driven, to some extent. To use the same example, this neighborhood summarization technique can convert a sentence that has been identified as an action item, "you need to do that before the next meeting", into a context-rich rephrasing, "Jake needs to fix the menu button on the website because our users are complaining that it does not work half the time."

We choose three sentences before and two sentences after for our neighborhood after experimenting with different values and inspecting the quality of the resulting summaries ourselves. Any smaller of a neighborhood, and we found that there was not enough context in the resulting summary. Any larger of a neighborhood, and the summary often did not revolve around the action item and instead addressed other parts of the input text that were not relevant for this particular action-item extraction task. It makes sense that we would need more sentences before the action item than after it since most pronoun references and necessary context would be provided before a sentence that depends on it. However, since this is a dialogue summarization task, and there are many anomalies when people speak, sentences after the action item are still necessary to include in the neighborhood in the event that additional pronoun references or context comes after. Note that there are edge cases, for example when an action item is located at the very beginning or end of a chunk, so please see Algorithm 2 for more details.

We append the action items with context from a given chunk to the end of the general summary for this same chunk. This way, we keep the summaries and action items that are derived from the same pieces of text together. Then we pass this entire text (summary + action items) into the same BART summarizer. We found that this technique helps condense the summary as well as improve the coherence of the resulting summary for each chunk.

3.4 Combining Summaries and the Recursive Case

Now that we have generated summaries for each chunk, containing information regarding both the general summary and the action items, we will generate an abstractive summary again based on all of the sectional summaries combined together in a recursive approach. If we append the sectional summaries together, and the number of tokens in this entire chunk of text is less than 1024, we pass this entire chunk of summaries into the same BART summarizer again; in essence, we are summarizing the summaries. However, if this entire chunk of summaries contains more than 1024 tokens, then we fall into the recursive case where we pass this entire chunk of summaries back into the entire function as if it is a meeting transcript. We explored other techniques to fluidly combine

[4]https://github.com/huggingface/neuralcoref

Algorithm 2 Action-Item Extraction(string text)

```
model ← action item classifier
tokenizer ← BERT tokenizer
actions ← empty string
sentences ← text split by sentence
for index, sentence in enumerate(sentences) do                ▷ Iterate through the sentences
    inputs ← tokenizer(sentence)
    predictedClass ← model(inputs)
    if predictedClass = 1 then                                ▷ Class 1 indicates sentence is an action item
        neighborhood ← empty string
        startIndex ← max(0, index − 3)
        endIndex ← min(len(sentences), index + 3)
        for neighborIdx in range(startIndex, endIndex) do
            neighborhood += sentences[neighborIdx]
        end for
        actions += generalSum(neighborhood)                   ▷ Summarize the neighborhood
    end if
end for
return actions                                                ▷ A string containing the context-rich action items found in text
```

the summaries together, but we found that using the BART summarizer achieved the best results. For example, we attempted to use an existing RoBERTa model [14] that was fine-tuned on a sentence fusion dataset known as DiscoFuse [19]. However, this technique did not prove effective because the resulting summaries were often very long and contained repetitions. We tried solving this problem by tuning the BART summarizer model to generate shorter sectional summaries, so the resulting chunk of all the summaries appended together would be shorter, but the sentence fusion models still did not prove effective in generating grammatically correct and coherent final summaries. This is a very challenging task if approached from a sentence fusion perspective, howerver, we approached this problem as simply another summarization task; the fine-tuned BART summarizer proved very effective at this task by removing repetitions between the sectional summaries and generating very informative, coherent, and concise summaries as seen in our results table.

4 RESULTS AND ANALYSIS

We first generated meeting summaries without including our action-item extraction technique in order to evaluate our three topic segmentation techniques and recursive algorithm. We evaluate within our own techniques as well as compare to the current state-of-the-art on the AMI dataset using the BART summarizer [20]. Then we compare our summaries with and without action items and show that our action-item-driven summaries contain additional valuable information.

4.1 Topic Segmentation Performance

We evaluate our topic segmentation methods by keeping our recursive algorithm constant and only varying the topic segmentation method. We see in Table 1 that all three of our novel topic segmentation methods outperformed linear segmentation with respect to both the BERTScore and ROUGE metrics. Most notably, with respect to the BERTScore metric, our methods, simple cosine segmentation, complex cosine segmentation, and chunked linear segmentation, outperform linear segmentation by 0.50% 1.07% and 1.36%, respectively for the generated summaries without action items. For the summaries with action items, the improvements over linear segmentation with respect to the BERTScore metric, were 0.38% 1.11% and 1.22%, respectively.

The complex cosine segmentation technique outperformed the simple cosine segmentation technique by 0.57% and 0.73% in terms of the BERTScore metric for the summaries without and with action items, respectively. This was expected because the former was less sensitive to "meaningless" turns as explained in the "Complex Cosine Segmentation" subsection. However, chunked linear segmentation, which does not rely on sentence embeddings and cosine similarity, outperformed all.

4.2 Recursive Algorithm Performance

We also compare the results of our recursive algorithm to the method proposed by Shinde et al. [20]. When we both use linear segmentation and the same fine-tuned BART models, but different "recursive" algorithms, our action-item-driven model outperforms the model presented by Shinde et al. [20] by approximately 4.98% in terms of the BERTScore metric. With regard to our general summarization model (without action items), this model still outperformed that presented by Shinde et al. [20] by approximately 4.77%. This means that, regardless of whether or not we include action items, the summaries our model generates are more similar to those of the human reference summaries in terms of their semantic meanings.

The model by Shinde et al. [20] does outperform our model in terms of the ROUGE scores, which measure lexical overlap, but this is expected since we use a truly recursive algorithm that results in the input text and the corresponding sectional summaries being passed into the BART summarizer more times. This would, of course, decrease the lexical overlap between our machine-generated

Algorithm 3 Action-Item-Driven Summary(string text, bool first, int maxTokens)

```
tokenizer ← tokenizer used by summarization model
if first = True then
    chunks ← topicalChunksBySpeaker(text)                ▷ Split text into topic-based chunks
else
    chunks ← topicalChunksBySentence(text)
end if
chunkSums ← array with size of len(chunks)
for all index ∈ range(0, len(chunks)) do                 ▷ Summarize each chunk in parallel
    part ← chunks[index]
    genSum ← generalSum(part)
    if first = True then
        actions ← actionItemExtraction(chunk)            ▷ Extract action items
        combined ← genSum + actions
        combinedNumTokens ← tokenLen(tokenizer(genSum + actions))
        if combinedNumTokens > maxTokens then            ▷ Theoretically possible but never true in our testing
            combined ← truncateText(combined)
        end if
        chunkSum ← generalSum(combined)
        chunkSums[index] ← chunkSum
    else
        chunkSums[index] ← genSum
    end if
end for
concatSums ← concatenate(chunkSums)                      ▷ Concatenate summaries after parallel loop completes
summaryNumTokens ← tokenLen(tokenizer(concatSums))
if summaryNumTokens > maxTokens then
    return actionItemDrivenSummary(concatSums, False, maxTokens)   ▷ Recursive call
else
    return generalSum(concatSums)                        ▷ The action-item-driven summary of text
end if
```

Topic Segmentation ↓ Metric →	BERTScore	R-1	R-2	R-L
General Summaries (Without Action Items)				
Linear Segmentation (Baseline Technique)	63.41	38.14	8.61	19.46
Chunked Linear Segmentation	**64.77**	**38.93**	**9.27**	**19.63**
Simple Cosine Segmentation	63.91	38.49	8.61	19.46
Complex Cosine Segmentation	64.48	38.92	9.24	19.47
Action-Item-Driven Summaries				
Linear Segmentation (Baseline Technique)	63.76	35.11	8.04	18.99
Chunked Linear Segmentation	**64.98**	**36.27**	8.31	**19.62**
Simple Cosine Segmentation	64.14	35.30	8.12	19.24
Complex Cosine Segmentation	64.87	36.21	**8.32**	19.61
Shinde et al., (2022)	60	45.2	13.3	-

Table 1: BERTScore and ROUGE evaluation scores for our machine-generated summaries across 4 different topic segmentation methods on the AMI corpus. This is done separately for both the general summaries (without action items) and the action-item-driven summaries. We also include the scores achieved by the current state-of-the-art model [20].

summaries and the human reference summaries. However, it seems that our summaries better match the semantic meaning of the human reference summaries, which was shown to be more important for human judgement by Zhang et al. [27].

4.3 Action-Item-Driven Summary Performance

As seen in Table 1, our action-item-driven summaries achieve slightly higher BERTScores than our general summaries (without action

General Summary (Without Action Items)	Action-Item-Driven Summary
Marketing Expert, Product Manager, and Industrial Designer are having a conceptual design meeting after lunch. They talk about the most important aspect for remote controls as people want a fancy look and feel. They discuss batteries, the design of the LCD display on the LCD screen, how to distinguish where people have to press the button when they have a flip-top, and how to incorporate voice recognition into the remote control. They agree on keeping the control buttons standardized and checking the financial feasibility. They decide to start with the black and white one and go for a whistle if financially voice recognition is not feasible. The product will have a logo on it just like everything else in a year's time if they get feedback from design fairs. Product manager will go through the end of the end meeting. Marketing Expert shares some information about a remote control that fits into the palm of the hand, made of plastic, with a rubberised cover, and the design is based on the input from the previous meeting.	Marketing Expert, Product Manager, and Industrial Designer are having a conceptual design meeting after lunch. They talk about properties, materials, user-interface and trend-watching. Marketing Expert says the fashion update which relates to very personal preferences among their subject group. There's no rechargeable option for the remote control, so they're going to look into battery options. Industrial Designer and Marketing Expert are talking about the size of the batteries they need to take into consideration. Marketing Expert thinks using the standard batteries and the solar charging will detract from the attractiveness of the whole feature. Marketing Expert thinks the buttons on the remote should have lights behind the buttons. Marketing Expert wants to make the basic mold out of plastic but have a rubber cover. Marketing experts are going to market to guys as much as to women. Marketing Expert shares with Industrial Designer some information about the design of the LCD display on the LCD screen. Industrial Designer and Marketing Expert are discussing how to incorporate voice recognition into the remote control. Industrial Designer tells Product Manager they need to get double A or triple A batteries. Sarah and Marketing Expert are talking about the design of a remote control with a rubberised cover. Industrial Designer tells Marketing Expert they can go for a whistle if voice recognition is not feasible. Product Manager will wrap up the end-of-meeting message.

Table 2: Comparison between machine-generated General (Without Action Items) and Action-Item-Driven Summaries when both methods employ chunked linear segmentation. The additions in the action-item-driven summary are underlined. AMI Meeting ID: ES2004c

items), but we consider this difference negligible (0.21% increase in BERTScore when both use chunked linear segmentation). However, we suspect that the reason for this small difference is that the human reference summaries in the AMI dataset appear to be more action-item-driven that those in the XSUM and SAMSUM datasets which we used to fine-tune our BART model.

The ROUGE scores for our action-item-driven summaries were notably lower than those achieved by our general summaries. For example, when both techniques employ chunked linear segmentation, the ROUGE-1 scores for our general summaries were 1.66% higher than those for our action-item-driven summaries. This makes sense since, in the action-item-driven summaries, we are deliberately adding words and phrases (action items) that are not included in the human reference summaries; thus, our precision score decreases. However, the slight increase in our BERTScores suggests that we are still capturing the semantic meaning of the reference summaries well.

Table 2 shows example outputs from our general model and our action-item-driven model when both algorithms employ chunked linear segmentation. We underline the additions in the action-item-driven summary and show that our action-item-driven model properly includes relevant action items from the meeting. Consider the following sentence from the action driven summary: "There's no rechargeable option for the remote control, so they're going to look into battery options." This action item is not included in either the general summary or the human reference summary, but it is a relevant and informative action item that adds value to the meeting summary. We also see that this action item is coherent and rich with context. This example and the other sentences underlined in Table 2 serve as evidence that our action-item extraction technique utilizing neighborhood summarization is quite effective.

5 FUTURE RESEARCH

In this study, we focused on generating action-item-driven summaries, but there are additional components of a good meeting summary. As noted in our "Introduction" section, decisions made, main topics, tension levels, etc. would also be very informative aspects of a meeting summary. While incorporating these elements into a meeting summary may lower our automated evaluation scores, this does not necessarily mean that the resulting meeting summary would be less useful for human readers. We hope to explore current approaches and develop new algorithms to extract these ideas from a meeting transcript and then incorporate them into a meeting summary.

While all three of our novel topic segmentation techniques outperformed linear segmentation, our best performance came from chunked linear segmentation, which did not involve calculating any embeddings or cosine similarities. However, the fact that chunked linear segmentation outperformed linear segmentation suggests that we can generate better summaries by minimizing the number of interrupted ideas in the meeting transcript. Thus, we hope to develop a more advanced topic segmentation method that will lead to better generated summaries and outperform chunked linear segmentation.

Finally, action-item extraction has not been explored in depth and has both a lack of techniques as well as metrics for evaluating these techniques. Thus, we hope to dive deeper into this issue and produce more advanced techniques for accomplishing the two above goals. Nevertheless, our neighborhood summarization algorithm proved very effective in action-item extraction, and we hope to test its performance on other tasks involving context resolution as well (e.g. extracting decisions made from a meeting).

6 CONCLUSION

This study explores a novel method for automatically generating meeting summaries by treating this problem as a fundamentally different one from that of generating dialogue summaries. Action items drive this recursively-generated, abstractive summary of the meeting that achieves approximately 4.98% higher BERTScores across the AMI corpus than the previous state-of-the-art using the BART summarizer. We introduce novel topic segmentation and action-item extraction algorithms that all improve and add value to the resulting summaries. The recursive approach presented in this paper for generating summaries for different parts and aspects of the meeting transcript can be expanded upon to improve meeting summarization, as well as be generalized and applied to summarizing other genres of text in the future.

REFERENCES

[1] Kevin Clark and Christopher D. Manning. 2016. Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 643–653. https://doi.org/10.18653/v1/P16-1061

[2] Amir Cohen, Amir Kantor, Sagi Hilleli, and Eyal Kolman. 2021. Automatic Rephrasing of Transcripts-based Action Items. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 2862–2873. https://doi.org/10.18653/v1/2021.findings-acl.253

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[4] Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. 2023. A Survey on Long Text Modeling with Transformers. http://arxiv.org/abs/2302.14502 arXiv:2302.14502 [cs].

[5] Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics* 9 (April 2021), 391–409. https://doi.org/10.1162/tacl_a_00373

[6] Xiachong Feng, Xiaocheng Feng, and Bing Qin. 2022. A Survey on Dialogue Summarization: Recent Advances and New Frontiers. http://arxiv.org/abs/2107.03175 arXiv:2107.03175 [cs].

[7] Mohammed Farooq Abdulla FM, Pawankumar S, Guruprasath M, and Jayaprakash J. 2022. Automation of Minutes of Meeting (MoM) using Natural Language Processing (NLP). In *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)*. 1–6. https://doi.org/10.1109/IC3IOT53935.2022.9767933

[8] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. 70–79. https://doi.org/10.18653/v1/D19-5409 arXiv:1911.12237 [cs].

[9] Som Gupta and S. K Gupta. 2019. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications* 121 (May 2019), 49–65. https://doi.org/10.1016/j.eswa.2018.12.011

[10] Marti A Hearst. 1997. TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages. *Computational Linguistics* 23, 1 (1997).

[11] Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. 2023. An Empirical Survey on Long Document Summarization: Datasets, Models, and Metrics. *Comput. Surveys* 55, 8 (Aug. 2023), 1–35. https://doi.org/10.1145/3545176

[12] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

[13] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. (2004).

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. http://arxiv.org/abs/1907.11692 arXiv:1907.11692 [cs].

[15] Iain Mccowan, J Carletta, Wessel Kraaij, Simone Ashby, S Bourban, M Flynn, M Guillemot, Thomas Hain, J Kadlec, V Karaiskos, M Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska Masson, Wilfried Post, Dennis Reidsma, and P Wellner. 2005. The AMI meeting corpus. *Int'l. Conf. on Methods and Techniques in Behavioral Research* (01 2005).

[16] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. http://arxiv.org/abs/1808.08745 arXiv:1808.08745 [cs].

[17] Ishmael Obonyo, Silvia Casola, and Horacio Saggion. 2022. Exploring the limits of a base BART for multi-document summarization in the medical domain. (2022).

[18] Virgile Rennard, Guokan Shang, Julie Hunter, and Michalis Vazirgiannis. 2023. Abstractive Meeting Summarization: A Survey. http://arxiv.org/abs/2208.04163 arXiv:2208.04163 [cs].

[19] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. *Transactions of the Association for Computational Linguistics* 8 (Dec. 2020), 264–280. https://doi.org/10.1162/tacl_a_00313 arXiv:1907.12461 [cs].

[20] Kartik Shinde, Tirthankar Ghosal, Muskaan Singh, and Ondr˘ej Bojar. 2022. Automatic Minuting: A Pipeline Method for Generating Minutes from Multi-Party Meeting Transcripts. (2022).

[21] Alessandro Solbiati, Kevin Heffernan, Georgios Damaskinos, Shivani Poddar, Shubham Modi, and Jacques Cali. 2021. Unsupervised Topic Segmentation of Meetings with BERT Embeddings. http://arxiv.org/abs/2106.12978 arXiv:2106.12978 [cs].

[22] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. http://arxiv.org/abs/2004.09297 arXiv:2004.09297 [cs].

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. http://arxiv.org/abs/1706.03762 arXiv:1706.03762 [cs].

[24] Jeff Wu, Long Ouyang, Daniel M. Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021. Recursively Summarizing Books with Human Feedback. http://arxiv.org/abs/2109.10862

[25] Atsuki Yamaguchi, Gaku Morio, Hiroaki Ozaki, Ken-ichi Yokote, and Kenji Nagamatsu. 2021. Team Hitachi @ AutoMin 2021: Reference-free Automatic Minuting Pipeline with Argument Structure Construction over Topic-based Summarization. http://arxiv.org/abs/2112.02741

[26] Xianjun Yang, Yan Li, Xinlu Zhang, Haifeng Chen, and Wei Cheng. 2023. Exploring the Limits of ChatGPT for Query or Aspect-based Text Summarization. http://arxiv.org/abs/2302.08081 arXiv:2302.08081 [cs].

[27] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. http://arxiv.org/abs/1904.09675 arXiv:1904.09675 [cs].

[28] Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed H. Awadallah, Dragomir Radev, and Rui Zhang. 2022. Summ^N: A Multi-Stage Summarization Framework for Long Input Dialogues and Documents. http://arxiv.org/abs/2110.10150 arXiv:2110.10150 [cs].

[29] Ming Zhong, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2022. DialogLM: Pre-trained Model for Long Dialogue Understanding and Summarization. http://arxiv.org/abs/2109.02492