UNIVERSITY OF CALIFORNIA,
IRVINE

Automated Extraction of Relevant Information in Software Maintenance Meetings: An Evaluation of Contemporary AI-based Summarization Approaches

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Software Engineering

by

Leyu Lyu

Thesis Committee:
Professor André van der Hoek, Chair
Professor Sam Malek
Assistant Professor Iftekhar Ahmed

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my committee chair, Professor André van der Hoek, for his invaluable guidance, unwavering support, and insightful feedback throughout this research journey. His expertise and encouragement have been instrumental in shaping this thesis and my growth as a researcher. Without his guidance and persistent help this thesis would not have been possible.

I would also like to thank my committee members, Professor Sam Malek and Professor Iftekhar Ahmed, for their time, constructive criticism, and valuable perspectives. Their input has significantly enhanced the quality and depth of this work.

Special thanks go to Professor Janet Burge of Colorado College and Adriana Meza Soria of IBM Research. Their prior research, dataset, and innovative ideas laid the foundation for this study. Their contributions to the field have been invaluable, and this thesis would not have been possible without their groundwork.

ABSTRACT OF THE THESIS

Automated Extraction of Relevant Information in Software Maintenance Meetings: An Evaluation of Contemporary AI-based Summarization Approaches

By

Leyu Lyu

Master of Science in Software Engineering

University of California, Irvine, 2024

Professor André van der Hoek, Chair

This thesis evaluates 16 meeting summarization approaches, including commercial tools and large language models (LLMs), across 45 software maintenance meeting topics. Assessing performance on truthfulness, essence, length, structure, and clarity, the results reveal significant variability among approaches. LLMs demonstrated more consistent performance than meeting tools, yet no single approach excelled in all metrics, highlighting inherent trade-offs in summarization. Notably, prompt design significantly impacted LLM performance. The study provides insights for tool selection and directions for future summarization approaches.

Chapter 1

Introduction

Communication is a key component for a successful software development team [12]. Meetings are one of the most effective mediums for disseminating information. Compared to other widely used communication channels, meetings can convey a substantial amount of information in a relatively short amount of time [1]. In the context of Global Software Engineering (GSE), projects are rarely resolved by a single team alone. Coordination in a multi-team GSE environment is challenging because the work is concurrently conducted by several development teams distributed across various sites [4]. Frequent and iterative delivery of results necessitates coordination at both the project and team levels. Therefore, meetings occupy a significant portion of time in the software development process.

Meetings are generally supported by a variety of tools that facilitate different aspects of the meeting process. Video conferencing platforms such as Zoom[1], Microsoft Teams[2], and Google Meet[3] enable audio and video communication. Document collaboration tools such as

[1] `https://zoom.us`
[2] `https://teams.microsoft.com`
[3] `https://meet.google.com`

Google Docs[4], Microsoft Office 365[5], and Dropbox Paper[6] allow for sharing and collaborating on documents. Screen sharing and virtual whiteboard tools, including Miro[7] and Mural[8], enhance visual interaction. Additionally, note-taking apps such as Evernote[9], OneNote[10], and Notion[11] facilitate efficient information capture during meetings. Meeting scheduling tools such as Calendly[12], Doodle[13], and Google Calendar[14] help coordinate participants' availability. Online conferencing and webinar tools such as GoToMeeting[15], WebEx[16], and Adobe Connect[17] support hosting larger virtual meetings and presentations.

During the COVID-19 pandemic, the usage of online meeting tools surged dramatically [11], driven by the widespread adoption of remote work and the need for virtual collaboration among distributed teams. Even as the pandemic subsides, many organizations have embraced hybrid or fully remote work models, recognizing the benefits of flexibility and cost savings [25]. This suggests that the reliance on online meeting tools is likely to persist and potentially increase further. In the context of GSE, where development teams are dispersed across different sites and time zones, the necessity for effective online meeting tools becomes even more pronounced. These tools play a crucial role in facilitating communication, coordination, and collaboration among geographically distributed teams, enabling real-time discussions, decision-making, and knowledge sharing vital for successful software development projects in a GSE environment.

The shift towards hybrid working environments—melding remote and co-located settings—has

[4] https://docs.google.com
[5] https://www.office.com
[6] paper.dropbox.com
[7] https://miro.com
[8] https://mural.co/
[9] https://evernote.com/
[10] https://www.onenote.com/
[11] https://www.notion.so/
[12] https://calendly.com/
[13] https://doodle.com/en/
[14] https://calendar.google.com/
[15] https://www.goto.com/meeting
[16] https://www.webex.com/
[17] https://www.adobe.com/products/adobeconnect.html

catalyzed a transformation in meeting tools. This new paradigm requires tools that not only bridge the gap between in-office and remote participants but also cater specifically to the unique dynamics of hybrid interactions. For example, platforms (e.g., Microsoft Teams, Zoom) have introduced features such as virtual breakout rooms, which allow smaller groups to collaborate within larger meetings, and real-time transcription services, ensuring that all participants can follow along regardless of background noise or connection issues. Additionally, tools such as Miro and MURAL offer virtual whiteboards that mimic in-office brainstorming sessions, fostering creativity and engagement among team members. These and other new features are aimed at optimizing user engagement and effectiveness, ensuring that all participants, whether dialing in from home or present in the meeting room, have equitable access to communication and resources.

Hybrid and meeting-intensive work environments increase the risk of missing critical information. If someone misses a meeting, catching up on the latest updates can become a time-consuming and arduous task. Additionally, taking detailed notes during lengthy meetings can be challenging. Traditionally, individuals had to read lengthy meeting reports or speak with attendees to understand the key points of a meeting. However, with the advancement of meeting tools, summary generation is now widely supported, which allows those who missed the meeting to quickly grasp the critical information, thereby saving time and effort. Consequently, summary generation has become one of the most important features of modern meeting tools.

Meanwhile, generative AI is making a profound impact across various industries, and these models can also be utilized to generate meeting summaries. Meeting tools are no longer the only option for generating summaries. By responding to different prompts, generative AI can create personalized meeting summaries tailored to individual needs.

Facing a diverse array of meeting summary generation approaches, including meeting tools and AI approaches, I pose the following research question: To what degree can these ap-

proaches automatically capture important and relevant information from software development meetings?

In this thesis, I conduct a comprehensive evaluation of the performance of 16 approaches across 45 meetings as distinct topics. The assessment encompasses a meticulous analysis of the generated summaries from five critical aspects: truthfulness, clarity, length, structure, and essence. By conducting this multifaceted evaluation, I seek to gain an understanding of the strengths and limitations of each approach, ultimately providing guidance and recommendations for the development of next-generation meeting tools.

Chapter 2

Literature Review

Meetings play a crucial role in the software development process, with development teams dedicating a significant amount of time to them. A quantitative study by Viktoria Stray et al. reveals that employees working on global projects spend an average of 7 hours and 45 minutes per week in scheduled meetings, and an additional 8 hours and 54 minutes in unscheduled meetings [24]. Meetings serve as a vital communication channel in the software development process, offering the ability to convey a substantial amount of information in a relatively short period compared to other commonly used communication methods [1]. This efficiency in information exchange is a key reason behind the significant time allocated to meetings throughout the software development lifecycle. The importance of meetings is further underscored by the findings of Schneider and colleagues [20], who have established a positive correlation between meetings and project success. Their research suggests that effective meetings lead to improved project outcomes.

Currently, various types of meeting tools provide support for meetings, with core features including transcription and summarization. Transcription, which involves the automatic conversion of spoken words into written text, provides a verbatim record of the meeting, al-

lows participants to focus on discussions rather than note-taking, and improves accessibility. Summarization condenses meeting content into concise overviews, saving time, highlighting key points and decisions, and facilitating easy sharing of outcomes. Both features contribute to improving meeting documentation, enhancing information retention, and streamlining communication [28].

Aside from transcription and summarization, some meeting tools also integrate additional functionalities to enhance the effectiveness of delivering information from speech transcript or summaries. As one example, timestamps in Otter.ai [19] are used to allow users to jump to specific locations in the transcript, directly linking to the sections where the summary's content is derived, so that a user can easily access and review the detailed context behind the summarized information. As another example, MeetGeek [17] cuts the meeting recording (video/audio) based on the importance of the content, producing a short, playable summary that retains only the key points. This allows users to quickly review the most important information without needing to watch or listen to the entire recording. Chatbots have also been integrated into some meeting tools [19, 7] to provide interactive information retrieval. That is, a user can ask for the information they need using natural language, and the responses they receive are entirely based on the content of the meeting. The report by J. Harjamäki et al. [10] provides an overview of the features, use cases, user reviews, and limitations of twelve current popular meeting tools from basic platforms that automate recording, transcription, and summarization to more advanced solutions offering unique features such as speaker time tracking, AI-driven assistance, and real-time audio-video enhancements.

Recent research shows LLMs have achieved significant advancements in summary generation. The experiment conducted by T. Goyal et al. indicates that summaries generated by a fine-tuned GPT-3 model are favored over those produced by a state-of-the-art summarization model, BRIO [9]. T. Zhang et al. [29] conducted a comparative analysis between LLM-generated and professional writer-crafted summaries, evaluating them on faithfulness,

coherence, and relevance, and found that state-of-the-art LLMs perform comparably to human freelance writers in summarization tasks. Although there are some limitations, for instance, according to Y. Zhou et al. [30], despite the considerable summarization abilities exhibited by ChatGPT and GPT-4, these models do not perform as well as those pre-trained for specific tasks like DialogSum (English social conversations) and DECODA (French call center interactions). As another example, a study conducted by Adithya Bhaskar et al. [2] demonstrates that, while GPT-3.5-based opinion summarization yields reviews that are highly fluent and coherent, they do not always accurately reflect the original input and may overly generalize specific viewpoints.

A key question that is important to all research in this area is how to assess the quality of generated summaries. To address this, various techniques have been proposed. Mani [13] and Dang [16, 3] evaluated the readability of a peer summary by answering questions in terms of a five-point scale to assess the readability of a summary. Building on earlier work, Teufel and van Halteren [26] introduced the Factoid method, which evaluates summaries by comparing overlaps of atomic information units called factoids. This approach extends Voorhees' [27] concept of information nuggets, using discrete facts to assess the content of machine-generated summaries. Ani Nenkova et al. [18] developed the Pyramid Method, an evaluation framework for summarization that employs a pyramid structure to represent key content units (SCUs) identified by multiple evaluators. These SCUs are organized hierarchically based on their selection frequency. Summaries are assessed by comparing their SCUs to the pyramid, with higher scores assigned to SCUs from upper layers, reflecting greater consensus on content importance.

The evaluation techniques above are applied manually, which is labor-intensive and time-consuming. Therefore, researchers have developed automated metrics designed to approximate human judgment. Early research in text such automated metrics adopted metrics from the field of information retrieval, including recall, precision, and F-measure, to quan-

titatively assess how much relevant information the approaches contain from the original document [15]. Although these metrics can be calculated automatically, they still rely on human summaries as references. ROUGE, introduced by Lin [14], is a suite of metrics for automatically assessing machine-generated summaries against human-authored references. It quantifies summary quality by measuring the number of overlapping units such as n-gram, word sequences, and word pairs between system-generated outputs and reference summaries, offering a standardized approach to evaluation in automatic text summarization research. Recently, H. Shakil et al. [21] utilized GPT as an evaluator instead of summarizer to independently assess the quality of summaries without relying on predefined metrics. Their results show that there is a significant correlation between GPT evaluations and traditional metrics, especially in assessing relevance and coherence.

Regardless of whether the summary evaluation is conducted manually or automatically, this task remains highly challenging due to the inherent subjectivity involved [6]. Considering the gaps between various evaluation protocols in recent papers, Alexander R. Fabbri et al. [5] combined automatic and manual evaluation to assess the current summary evaluation metrics. Their results highlight the disparities in the understanding of evaluated dimensions between human evaluators and demonstrate that currently available automatic metrics do not reliably cover these dimensions.

Overall, while extensive research has focused on summary evaluation, few studies have specifically addressed software maintenance meetings [23, 22], and none have combined this focus with an analysis of both meeting tools and LLMs. This thesis contributes to the field by conducting a comparison of current meeting tools and LLMs in the context of summarizing software maintenance meetings. By examining their performance across multiple aspects, this thesis provides insights into the effectiveness of meeting tools and LLMs in capturing important information from the software maintenance meeting discussions.

Chapter 3

Method

I conducted a comparison experiment among eight meeting tools and two LLMs (hereafter referred to as "approaches"), applying them to 45 topics from 10 software maintenance meetings, and then manually grading the generated summaries based on the criteria of truthfulness, essence, length, structure, and clarity.

3.1 Dataset

The dataset used in this thesis stems from Adriana Meza Soria et al. [23]. It consists of ten software maintenance meetings held by an architecture committee from a major healthcare software provider. Soria's team coded these ten meetings and divided them into 45 topics. The dataset includes the recordings and human transcripts of the meetings.

3.2 Experiment Setting

As shown in Table 3.1, I selected eight widely adopted meeting tools based on software marketplace performance evaluated by a data analysis website G2 [8] in August 2023. In addition, I included two LLMs in the assessment for the experiment. Based on the usage scenarios and to leverage the potential of LLMs, each is configured with four prompts, as shown in Table 3.2. Naive Use simulates a user generating a meeting summary with an LLM without complex requirements. ADR Based instructs the LLM to capture information using Architecture Decision Records (ADR). Criteria Based involves guiding the LLM according to the five criteria mentioned in the previous section. Essence Based, on the other hand, directs the LLM to answer the core elements of the meeting as outlined in Section 3.3.2.

Approaches	**Meeting Tools / LLMs**	**URL**
Meeting Tools	Deepgram	`https://www.deepgram.com`
	Fireflies	`https://www.fireflies.ai`
	Krisp	`https://krisp.ai`
	MeetGeek	`https://meetgeek.ai`
	Otter.ai	`http://www.otter.ai`
	Sembly AI	`https://www.sembly.ai`
	Speechmatics	`http://www.speechmatics.com`
	Zoom AI	`https://www.zoom.com/en/ai-assistant`
Large Language Models (LLMs)	Claude3 (Naïve Use)	`https://www.anthropic.com/api`
	Claude3 (ADR Based)	`https://www.anthropic.com/api`
	Claude3 (Criteria Based)	`https://www.anthropic.com/api`
	Claude3 (Essence Based)	`https://www.anthropic.com/api`
	GPT-4 (Naïve Use)	`https://platform.openai.com`
	GPT-4 (ADR Based)	`https://platform.openai.com`
	GPT-4 (Criteria Based)	`https://platform.openai.com`
	GPT-4 (Essence Based)	`https://platform.openai.com`

*The content in parentheses indicates the type of prompt used for configuration.

Table 3.1: The Approaches Evaluated in the Project

Naïve Use	**ADR Based**
Generate a short meeting summary for transcript (With Markdown Format): • You can ignore minor, inconclusive discussions or small talk. • Try to use headings and bullets or other structure to make it clearer.	Using Architecture Decision Record, summarize the meeting transcript (with Markdown format).
Criteria Based	**Essence Based**
Generate a short summary for the meeting transcript (with Markdown format): • Ensure the Truthfulness of the summary. • You can ignore minor, inconclusive discussions or small talk. • Try to use multi-level headings and bullets or other structure to make it clearer. • Try to capture decisions and tasks. If possible, list their relevant members and deadline. • The summary should not be too long; you need to make trade-offs. • Ensure the clarity of the summary.	Try to answer the questions below based on the meeting transcript (using Markdown format): 1. What is the issue/problem being discussed? 2. Why does it happen/what causes it/how did it happen? 3. What is the solution/resolution? [What do we (plan to) do about it?] 4. Why do we choose this solution/resolution (and not something else, if considered)? 5. What are concrete action items? 6. Action executed during the meeting (as short as possible).

Table 3.2: Prompts applied in LLMs

3.3 Summary Collection

As shown in Figure 3.1, during the summary collection process, meeting recordings were directly imported into meeting tools in MP3 audio format. Since LLMs do not inherit transcription functionality on the web, and better reflect a typical user's experience, the official transcribe APIs of GPT and Claude were not utilized. Instead, the audio was converted into transcripts by the transcribe tool Rev and then imported into LLMs in text form.

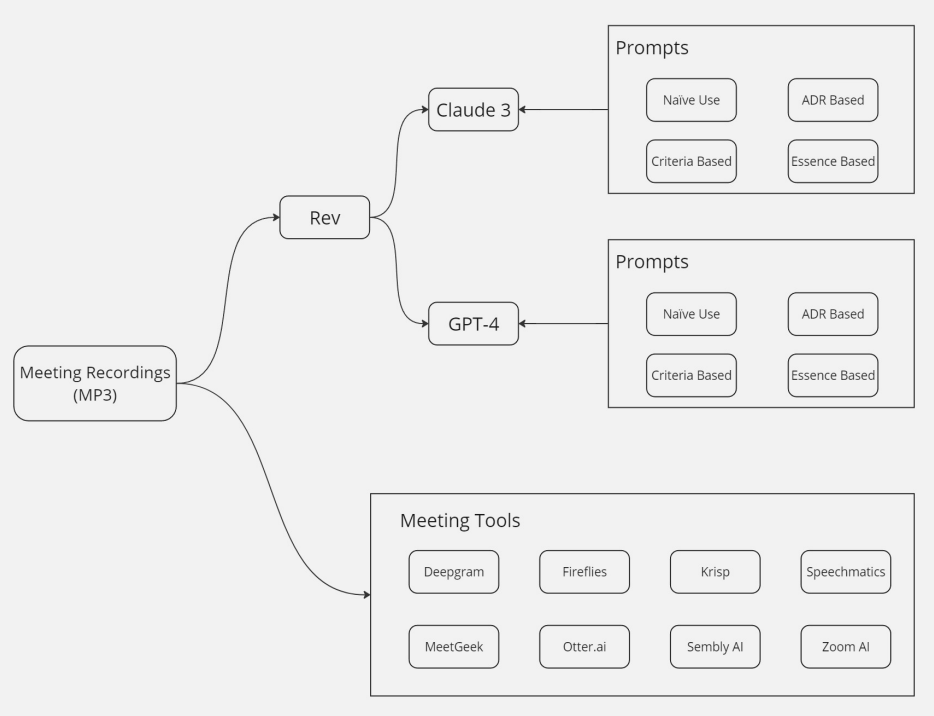

Figure 3.1: The Process of Summary Collection

All summaries generated by the meeting tools were completed before December 2023. The Naive Use, ADR Based, and Criteria Based prompts were utilized with GPT-4-1106-preview, while the Essence Based prompt was applied with GPT-4-turbo-preview (0125). All prompts were configured on Claude-3-opus-20240229.

3.4 Criteria

The generated summaries were evaluated according to five criteria: truthfulness, essence, length, structure, and clarity. Each criterion was assessed on a five-point scale to provide a quantified evaluation of each summary's quality.

Compared with existing research, Coherence is not included in the assessment. This decision is based on observations that most summaries are not composed of multiple continuous paragraphs, but rather consist of various levels of headings, bullets, and short paragraphs. Therefore, Coherence does not effectively reflect the quality of a summary. Instead, I introduced the criteria of structure and clarity. The former measures the degree of organization within the summary, while the latter assesses whether sentences within the summary are clearly expressed.

3.4.1 Truthfulness

The truthfulness of meeting summaries is of utmost importance for software developers, especially when they may not have attended the meeting or have forgotten some of the discussed information.

Truthful summaries allow developers to confidently rely on the information provided without the need for repeated verification. On the other hand, inaccurate or misleading summaries can have serious repercussions for software developers and the project. If a summary distorts the information discussed in a meeting, developers may make decisions or take actions based on incorrect assumptions, leading to wasted effort, delays, or even the introduction of errors in the software. This could potentially result in increased development costs, missed deadlines, and damage the team's morale and trust in the communication process. To assess truthfulness, I focus on factual errors. The grading standards for truthfulness are as follows

(Table 3.3):

Grade	Criteria
5	No factual error
4	1 factual error
3	2 factual errors
2	3 factual errors
1	4 factual errors
0	5 or more factual errors
F	Failed to generate a summary

Table 3.3: Grading Criteria for Summary

The maximum grade is 5 points. For each factual error, one point is deducted, down to a minimum of 0 points. For example, consider the following information:

> Developer 1: "I think we've all agreed that Plan A is the best solution so far."
> Developer 2: "Yeah, it seems solid. But we still need to check in with the external team before we finalize anything."
> Developer 3: "Absolutely. We can't move forward without their input. We should communicate that to them as soon as possible."
> Developer 1: "Good point. Let's make sure they're on board before we proceed."

An approach summarizes it as,

> "The team considers Plan A the best solution, pending external communication."

The statement is accurate as it correctly reflects the team is ongoing deliberation and the fact that external confirmation is still required. In contrast, another approach summarizes this as,

> "The team has confirmed Plan A as the solution."

It introduces a factual error because it inaccurately suggests the decision has been finalized. The summary would mislead readers into thinking no further steps are needed.

During the grading process, each summary is segmented into individual sentences using periods or line breaks. Each sentence is then compared with the original transcript by one grader. Any sentence that contradicts the transcript or includes content not mentioned in it is identified as a factual error.

In the context of GSE, where meeting participants often come from different countries, transcribing names can be particularly challenging. Additionally, software development meetings frequently involve specialized terms or abbreviations. Therefore, spelling mistakes in such names or terms that do not lead to ambiguity are not regarded as factual errors. For example, if "FTP" is mistakenly transcribed as "FDP," but "FDP" is not mentioned in the meeting and is not a commonly used term, it does not create ambiguity and thus is not considered a factual error. However, if the name "Angela" is recognized as "Angular," which refers to a web development framework, and this technological term also appears in the discussion, then this spelling mistake would be regarded as a factual error.

3.4.2 Essence

I use essence to assess the extent to which generated summaries capture the key information discussed in a meeting. Software development meetings often involve rejected proposals, extensive discussions, and casual conversations that may be unrelated to the final outcomes. An effective summary should focus on the essential points and disregard irrelevant information. The primary purpose of a meeting summary is to enable developers to quickly review the entire meeting. By focusing on the essence, the summary ensures that only the most critical and relevant information is included, providing a clear and concise representation of the meeting's important points.

I abstracted the core content of the meetings by considering the following six questions:

1. **"What is the issue/problem being discussed?" (Issue):** This question aims to identify the core problem or challenge that the meeting is addressing, which provides context for the rest of the discussion. By clearly stating the problem, the summary helps developers understand the focus of the meeting and the importance of the subsequent points.
2. **"Why does it happen/what causes it/how did it happen?" (Cause):** Understanding the root cause of the topic being discussed is crucial for developing an effective solution. The summary should include information on the factors contributing to the problem, such as underlying technical issues, process inefficiencies, or external circumstances.
3. **"What is the solution/resolution? / What do we (plan to) do about it?" (Solution):** Once the problem and its causes have been established, the meeting summary should outline the proposed solution or resolution. This question ensures that the summary includes a clear description of the steps or actions that the team has decided to take to address the issue. By highlighting the agreed-upon solution, developers can quickly grasp the plan of action and align their efforts accordingly.
4. **"Why do we choose this solution/resolution (and not something else, if considered)?"** (Rationale): In some cases, multiple potential solutions are discussed during the meeting. The rationale behind the chosen solution should be included in the summary to explain why it was selected over other alternatives. By providing this justification, developers can understand the thought process behind the decision and feel more confident in the chosen course of action.
5. **"What are concrete action items?" (Action):** To ensure that the

solution is implemented effectively, the meeting summary should clearly outline the specific tasks or action items assigned to team members. By breaking down the solution into concrete action items, the summary helps developers understand their individual responsibilities and contributes to a more organized and accountable project workflow.

6. **"What actions have been executed during?" (Done):** This question prompts the summary to include actions that were taken during the meeting. By including this information, the summary keeps developers informed about the project's advancement.

During the grading process, one person answered the above questions based on the original meeting transcript, providing a human-generated reference for comparison. Another person then evaluated the generated summary against the human-generated answers to determine how well the summary addressed each question. The essence criterion was assessed using the following scale:

- The grade for Essence ranges from 0 to 5 points.
- If a question is completely unanswered in the generated summary, 1 point is deducted from the total score.
- If a question is only partially answered in the generated summary, 0.5 points are deducted from the total score.

For example, the human-generated reference summary of Topic 7 is shown as follows:

1. **"What is the issue/problem being discussed?" (Issue):**
   Additional clients will possibly be onboarded in the upcoming months. Will that cause any issues with NG workflows in terms of the load on the server?

2. **"Why does it happen/what causes it/how did it happen?" (Cause):**
   One concern related to onboarding new clients is that the size of the client's DB may increase. The server already has a 90% load at times, with some spikes, and it is a shared server among multiple clients.
3. **"What is the solution/resolution? / What do we (plan to) do about it?" (Solution):**
   The solution would be to acquire additional "computational/server capacity" [at Amazon].
4. **"Why do we choose this solution/resolution (and not something else, if considered)?" (Rationale):**
   There is only a 5-to-15-minute downtime for upgrading the database.
5. **"What are concrete action items?" (Action):**
   Monitor the load on the server to ensure it is okay as other clients are brought on board.
6. **"What actions have been executed during?" (Done):**
   None

The summary generated by Speechmatics is shown in Figure 3.2. For the first question (Issue), the summary did well by identifying the potential onboarding of more clients and its impact on the server load, which was basically covered. For the second question (Cause), it partially addressed concerns about database size and spikes but missed mentioning the existing 90% server load and shared environment, resulting in a partially answered grade. The third question (Solution) was also partially answered as the summary focused on optimizing spikes instead of explicitly mentioning acquiring additional computational/server capacity. The fourth question (Rationale) was not mentioned since the summary did not address the rationale behind the solution, particularly the minimal downtime for database upgrades. For the fifth question (Action), the summary basically covered the need to monitor the server

load as more clients are onboarded. Lastly, the sixth question (Done) was basically covered as the summary correctly reflected that no actions were executed during the meeting. Therefore, the summary below is graded as 3 in total.

Speechmatics

Key Topics:
- Share workflows
- Impact on other clients
- Onboarding more clients
- Increase in database size
- Cost difference tracking
- Compute usage
- Spike optimization
- Monitoring load

Discussion:
- Share workflows are smooth sailing with no real issues impacting other clients.
- Developer D Solutions owner plans to onboard more clients, potentially doubling the number of calls into HGH.
- Concerns about the increase in database size and the associated downtime and cost.
- Developer Y wants to track the cost difference between when workflows were live and when they weren't.
- Costs won't change as the size of the database remains the same.
- Increase in transactions during the month of March can be confidently attributed to the workflows.
- The system S is also turning on feeds, and SC's system is ramping up.
- Compute usage is below 20% with momentary spikes.
- Optimizing spikes can prevent the need to increase the database size.
- Graphs show CPU usage and database costs for different clients.
- Load is expected to be fine if the number of calls is doubled, but monitoring is necessary.
- Service E cluster spiked, initially thought to be an alert for Postgres.

Figure 3.2: A summary of Topic 2 Generated by Speechmatics

3.4.3 Length

One of the primary objectives of summarization is to save time, which means that meeting tools must provide all key information without being overly lengthy or detailed. It is important to balance covering the essence with maintaining a concise length. If the focus is solely on capturing the essence without considering the length, the summary may become too detailed, resembling the original transcript and defeating the purpose of saving time.

Therefore, an ideal summary should prioritize critical content, ensuring that it is both concise and complete. I introduce a formula Word Count Ratio (1) to evaluate the performance of different approaches in terms of length:

$$\text{WCR} = \frac{\text{Number of Words in the Generated Summary}}{\text{Number of Words in the Original Transcript}} \tag{1}$$

This metric provides a quantitative measure of the summarization ratio, indicating how concisely the summary captures the content of the original transcript. A lower WCR value indicates a more condensed summary, meaning that more of the original content may be compressed or omitted. To further compare the performance of approaches in terms of length, as shown in Table 3.4, I use the following values:

Grade	**Criteria**
5	$\text{WCR} \in (0, 0.15]$
4	$\text{WCR} \in (0.15, 0.25]$
3	$\text{WCR} \in (0.25, 0.35]$
2	$\text{WCR} \in (0.35, 0.5]$
1	$\text{WCR} \in (0.5, 1)$
0	$\text{WCR} \in [1, +\infty)$
F	Failed to generate a summary

Table 3.4: The criteria of Length

3.4.4 Structure

A summary's structure plays a crucial role in its effectiveness. A well-organized summary with a logical flow of information enhances readability and comprehension. Incorporating structural elements such as headings and bullet points can improve a summary's clarity and accessibility. This is particularly relevant in software development meetings, where discussions can encompass multiple subtopics within a main topic. Moreover, these kinds of meetings frequently result in specific actions or decisions that warrant distinct emphasis.

By clearly delineating subtopics and highlighting actions or decisions, the summary can more effectively serve its practical purpose. Therefore, an optimal summary should not only condense information, but also present it in a structured, easily digestible format that facilitates quick and efficient review by the development team.

To evaluate the structural elements in generated summaries, I use the following criteria (Table 3.5). Structural elements include, but are not limited to, multiple paragraphs, bullets, headings, and multiple-level headings. Structural conjunctions are not considered structural elements, but words such as “Firstly” and “What’s more” can still enhance readability, especially when all content is compressed into a single paragraph. Therefore, structural conjunctions serve as a factor to differentiate between a score of 0 and 1.

Grade	**Criteria**
5	Use at least three structural elements in addition to multiple paragraphs
4	Use two structural elements in addition to multiple paragraphs
3	Use only one structural element in addition to multiple paragraphs
2	Use only multiple paragraphs
1	Consists of a single paragraph with structural conjunctions
0	Comprises only one paragraph without structural conjunctions
F	Failed to generate a summary

Table 3.5: The Criteria of Structure

For instance, the summary shown in Figure 3.3, is composed of multiple paragraphs and utilizes multi-level headings. Additionally, it employs bullets to separately annotate ”Actions.” Based on the above, the structure of this summary is graded as 5 points.

Meeting Summary

In the meeting, there was a discussion about documentation and training for AWS. It was decided that to access the prod databases, one needs to go to RDS. A task was assigned to set up a bastion account and request it on the channel, with Developer B responsible for updating the list of names. Another topic discussed was creating support tickets for assistance, and a task was assigned to check with Shabbat again for the file. Lastly, there was a discussion about requesting a production admin role for Developer S, and it was mentioned that if the team doesn't have time to fix the cursor problem, the team I will handle it tomorrow.

Next steps

- Set up a bastion account and request it on the channel. Developer B needs to update the list of names.
- Check with Shabbat again to see if he has the file.

Highlights

1. **Documentation and Training** (4:2)
   Discussion about documentation and training for AWS
   - **Action** | To access the prod databases, you need to go to RDS and then you will be able to view all of them.

2. **Requesting production admin role for Developer S** (0:54)
   Discussion about requesting production admin role for Developer S
   - **Fact** | If the team doesn't have time to fix the cursor problem, the team I will handle it tomorrow.

Figure 3.3: A 5-point Summary Generated by MeetGeek

3.4.5 Clarity

An optimal summary should exhibit clarity, eschewing ambiguity in its presentation. In an ideal scenario, readers should be able to grasp the main points and pertinent details without the necessity of re-reading or referencing the original transcript. This level of clarity is crucial for ensuring that the summary serves its purpose effectively, allowing developers to quickly assimilate the essential information from the meeting without requiring additional time or effort for interpretation or clarification. To assess the clarity of generated summaries, I note any area that appears ambiguous or unclear during the reading process. These instances of ambiguity or lack of clarity are defined as "reading obstacles". For example, one bullet point states:

"- The team decided to approve the design changes, ensuring that the original

concept was maintained."

This statement is ambiguous because it suggests both approval of changes and maintaining the original concept. The reader might be confused about whether the design changes alter the original concept or are meant to preserve it. This semantic ambiguity could lead to different interpretations of the team's decision and would require revisiting the original discussion or document to clarify.

As shown in Table 3.6, each identified reading obstacle results in a one-point deduction, down to a minimum score of 0 points.

Grade	**Criteria**
5	No reading obstacle
4	1 reading obstacle
3	2 reading obstacles
2	3 reading obstacles
1	4 reading obstacles
0	5 or more reading obstacles
F	Failed to generate a summary

Table 3.6: The Criteria of Clarity

Chapter 4

Result

The results are presented separately for each criterion, providing a detailed analysis of the performance of each of the approaches across the five aspects of summarization quality. Following individual analyses, I present an overall analysis to assess the comprehensive summarization capabilities of each approach.

4.1 Truthfulness

As depicted in Table 4.1, all approaches demonstrated exceptional performance in Truthfulness. While Deepgram's mean grade was slightly lower at 4.64, the other tools all received mean grades exceeding 4.9, i.e., there is no more than one factual error in a certain summary. This indicates that the current selected approaches have demonstrated proficiency in accurately capturing and conveying information from the original transcript.

Model	**Mean**	**Median**	**Mode**	**Max**	**Min**	**Counts (F)**	
Claude 3 (Naïve Use)	4.96	5	5	5	5	0	
Claude 3 (ADR Based)	4.96	5	5	5	5	0	
Claude 3 (Criteria Based)	4.96	5	5	5	5	1	
Claude 3 (Essence Based)	4.96	5	5	5	5	0	
GPT-4 (Naïve Use)	4.91	5	5	5	4	0	
GPT-4 (ADR Based)	4.96	5	5	5	5	0	
GPT-4 (Criteria Based)	5.00	5	5	5	5	0	
GPT-4 (Essence Based)	5.00	5	5	5	5	0	
Deepgram	4.64	5	5	5	3	0	
Fireflies	4.96	5	5	5	5	0	
Krisp	5.00	5	5	5	5	11	
MeetGeek	4.91	5	5	5	5	16	
Otter.ai	4.96	5	5	5	4	16	
Sembly AI	5.00	5	5	5	5	4	
Speechmatics	4.91	5	5	5	5	1	
Zoom AI	4.91	5	5	5	5	0	

Table 4.1: The Statistical Overview of Truthfulness Criteria Analysis

For example, in Topic 1, the software maintenance team are confirming whether their AWS account has access to the bastion. Some developers confirmed their access, but some developers need to create a ticket to request access. In Sembly AI, the discussion is summarized as:

2. Role Assignments and Access Testing • 00:02:32

   - Verification of operator admin access and need for admin access.

- Suggestion to test role assumption and report back.
- Request to create support ticket for admin access with Developer S's approval.

Although the summary may not include all the details, there is no factual error. However, in GPT-4 (Naïve Use), it is summarized as:

B2C Team Access

- **Amazon Accounts**: It was confirmed that B2C team has their Amazon accounts set up.
- **AWS Prod Access**:
- A follow-up is required to confirm if the B2C team has AWS prod and batching access.
- At least two individuals need this access to support migration.

Evidently, the underlined bullet point contains a factual inaccuracy. It incorrectly states that the Amazon account setup was completed, when in fact this task remains unfinished.

Notably, the summaries generated by different approaches vary significantly in their level of detail, which may introduce an element of inequity in the application of this criterion. For instance, consider following two summaries of the same discussion. Otter.ai records it concisely as:

"Workflows and database size impact on clients."

In contrast, Sembly AI provides a more detailed account:

Impact of onboarding more clients to Ng share workflows

- A developer reached out about onboarding more clients in the next few months, potentially doubling the number of calls.
- Concerns raised about the size of the database and potential impact on performance.
- Discussion on increasing the size of the database and associated costs.

Evidently, the former summary provides minimal detail and makes fewer specific claims, while the latter includes more detailed information. As a result, although neither contains factual errors, it is in a way easier for tools that produce more concise summaries, since there are fewer claims and details that possibly get wrong.

In the grading process, despite the method in Chapter 3 stating that all non-confusing personal names and technical terms with spelling errors were to be disregarded, misidentification of these terms frequently occurred in the dialogues, particularly for names that are not native to English.

4.2 Essence

As shown in Table 4.2, the overall mean grade for essence across all evaluated approaches is 3.44. There is a significant performance disparity between LLMs and meeting tools in capturing the essence of discussions. LLMs demonstrate better performance with a mean grade of 3.90, outperforming meeting tools, which achieved a mean grade of 2.98.

Model	**Mean**	**Median**	**Mode**	**Max**	**Min**	**Counts (F)**	
Claude 3 (Naïve Use)	3.93	4	5	5	2	0	
Claude 3 (ADR Based)	3.70	4	4	5	2	0	
Claude 3 (Criteria Based)	3.80	4	5	5	1.5	1	
Claude 3 (Essence Based)	4.07	4.5	5	5	2.5	0	
GPT-4 (Naïve Use)	3.97	4	4.5	5	2.5	0	
GPT-4 (ADR Based)	3.83	4	3.5	5	2.5	0	
GPT-4 (Criteria Based)	3.77	4	4.5	5	2	0	
GPT-4 (Essence Based)	4.10	4	4	5	2.5	0	
Deepgram	2.07	2	1.5	4	0	0	
Fireflies	3.90	4	4	5	2.5	0	
Krisp	3.20	3	3	5	1	11	
MeetGeek	1.87	2.5	2.5	4.5	0.5	16	
Otter.ai	2.30	2.5	2.5	5	1	16	
Sembly AI	3.47	3.5	3	4.5	2	4	
Speechmatics	3.70	3.5	3.5	4.5	2.5	1	
Zoom AI	3.37	3	2.5	4.5	1.5	0	
LLMs	3.90						
Meeting Tools	2.98						
Overall	3.44						

Table 4.2: The Statistical Overview of Essence Criteria Analysis

Among LLMs, Claude-3 configured with the essence-based prompt demonstrated the highest performance, achieving a mean grade of 4.07. Conversely, Claude-3 (ADR Based) exhibited the lowest performance within the LLM category, with a mean grade of only 3.70. In addition, all LLM approaches achieved scores exceeding the overall mean score across all approaches

(3.44).

The impact of prompts on LLM performance is noticeable. The essence-based prompt consistently outperformed other prompts in both GPT-4 and Claude-3 models. This essence-based prompt, comprising six questions designed to elicit key information, demonstrates that LLMs perform more effectively when explicitly instructed on the type of information to capture. This finding suggests that other prompts have substantial potential for improvement. Interestingly, both in Claude-3 and GPT-4, the criteria-based prompt yielded poorer performance compared to the naïve use approach. This observation suggests that providing LLMs with more complex and specific prompts may, in some cases, adversely affect their performance. This phenomenon potentially indicates a limitation in LLMs' ability to effectively process and act upon intricate prompts.

Within meeting tools, there is a significant variation in performance scores. Fireflies stands out with an exceptionally high score of 3.90, making it the only meeting tool to surpass the mean grade across all approaches. MeetGeek exhibits the lowest performance among all approaches with a score of only 1.87, followed by Deepgram and Otter.ai, which received grades of 2.07 and 2.30, respectively. The remaining meeting tools demonstrate a more clustered performance, with scores predominantly falling within a narrower range of 3.20 to 3.70.

The following example compares a high-quality summary and a low-quality summary. For reference, the human-generated summary against which both are compared is as follows:

1. **"What is the issue/problem being discussed?" (Issue):**
   The entire system was being clogged by one tenant, Health Current, causing another tenant, Santa Cruz, to be backed up by about 12 hours. It was responding, but very slowly (30 seconds).
2. **"Why does it happen/what causes it/how did it happen?" (Cause):**

Because both tenants (clients) share the same computing layer, one client can negatively affect another. The client did not have source queueing on, which is best practice in this case. Because responses were very slow (30 seconds or longer), the system was backing everything up.

3. **"What is the solution/resolution? / What do we (plan to) do about it?" (Solution):**

   Separate evaluation queue from action queue, and create action queues per destination/client system. Failure may occur, for which it is proposed to include automatic retry based on the error table that is already present. Advisory locks of the Rabbit queue are suggested to implement back pressure, so clients are responsible for their own back pressure handling.

4. **"Why do we choose this solution/resolution (and not something else, if considered)?"** (Rationale):

   Guaranteeing quality of service to all clients was mentioned as an alternative, but considered too complicated. Not retrying everything was also mentioned as an alternative. They talked about using lambdas, but considered that too expensive a solution. Another solution would be to delegate to Mirth Connect, but that is considered not good self-defense programming.

5. **"What are concrete action items?" (Action):**

   Put POC (proof of concept) on next Baggle agenda. Create some tickets to document the steps to be taken, with the first ticket being splitting off the action sender service.

6. **"What actions have been executed during?" (Done):**

   Not mentioned

The summaries generated by MeetGeek and Claude 3 (Criteria Based) are presented in Figure 4.1 and Figure 4.2 respectively. MeetGeek received a grade of 2 and Claude 3 (Criteria

Based) received a grade of 4.5. MeetGeek's summary partially answered some questions while Claude 3 (Criteria Based)'s summary basically covered all the information provided in the human summary. For the first question, MeetGeek only touched on the issue of resource allocation but does not mention the specific problem of one tenant causing a backup for another tenant, while Claude 3 (Criteria Based) clearly pointed out that "System H's slow external system (32 second responses) clogged the shared compute layer, causing delays for System S". For the second question, MeetGeek addressed the general cause (concurrency and resource allocation), but did not mention the specific reason related to shared computing layers and lack of source queueing. However, Claude 3 (Criteria Based)'s summary mentioned the shared compute layer and suggested creating a queue to isolate its impact. For the fourth and fifth question, MeetGeek did not explicitly mention the rationale behind the chosen solution and action items. In contrast, Claude 3 (Criteria Based) explained why other possible solutions were not chosen and implicitly listed the action item. This comparison clearly demonstrates that the summary with the higher essence grade captured more critical information from the original meeting.

As detailed in Table 4.3, LLMs performed better than meeting tools on all questions, and the trends between LLMs and meeting tools were similar. All approaches excelled in Done, indicating that their generated summaries could accurately capture what issues were discussed. Conversely, the approaches performed worst on Cause and Rationale, with mean grades lost as high as 0.41 and 0.34, respectively. This indicates that they have limitations in summarizing the causes of issues and the reasons for choosing solutions. The approaches also showed relatively low proficiency in summarizing Action, with an average grade lost reaching 0.26.

MeetGeek

Meeting Summary

In the meeting, there was a discussion about implementing back pressure and limiting concurrency for each external system globally. The concern was that if there is a flood of messages to send to an external system, all available threads are used up, so there is a need to limit concurrency to ensure proper resource allocation. It was suggested to use Redis or Postgres advisory locks to achieve this, where a message is pulled off the action queue and an advisory lock is obtained for the external system. If the lock cannot be obtained, the message is dropped. Another topic discussed was the use of Postgres advisory locks to ensure only one service can service one external system at a time. Additionally, there was a proposal to create one queue per external system and bind services to the relevant queues, with the emphasis on taking small steps to improve the situation.

Highlights

1. Back pressure and limiting concurrency for external systems (2:57)

Discussion on implementing back pressure and limiting concurrency for each external system globally.

- Concern | The reason we need to implement this on our side is because currently, if we receive a flood of messages to send to external system A, all of our available threads are used up, so we want to limit the concurrency for each external system to ensure proper resource allocation.
- Fact | We can achieve this by using Redis or Postgres advisory locks, where a message is pulled off the action queue and an advisory lock is obtained for the external system. If the lock cannot be obtained, the message is dropped in the table.

2. Using Postgres advisory locks for managing external system concurrency (3:1)

Exploration of using Postgres advisory locks to ensure only one service can service one external system at a time.

- Concern | The speaker believes that it is not practical to modify the PDF CDA content at this point in time, as mentioned earlier in the discussion about having a service for correctly managing the content.

3. Creating one queue per external system (4:31)

Proposal to create one queue per external system and bind services to the relevant queues.

- Important | Taking small steps is crucial in order to improve the situation and make it better.

Figure 4.1: A Summary of Topic 3 Generated by MeetGeek

MeetGeek

Performance Issues with Automated Workflow Subsystem
- System H's slow external system (32 second responses) clogged the shared compute layer, causing delays for System S
- Need a plan to better protect and guarantee quality of service for each environment

Proposed Solutions
1. Separate action queue from evaluation queue
   - Allow evaluations to continue even if actions are blocked
2. Create a queue for each external system destination
   - Isolate the impact of slow destinations
   - Challenges with deployment complexity and scalability
3. Apply back pressure or exponential backoff for slow endpoints
   - Delay sending the next message if an external system takes too long to respond
   - Requires keeping state in memory (e.g., Redis)
4. Implement timeouts and retries
   - Define reasonable timeout values
   - Retry failed messages, with some exceptions (e.g., authentication failures)
5. Use Postgres advisory locks for concurrency control
   - Only one service can process an external system at a time
   - Avoids the need for complex back pressure mechanisms

Action Items
- [ ] Create tickets for splitting the evaluator and action sender into separate services
  - Assignee: TBD
  - Deadline: Next sprint

Additional Concerns
- Scalability issues with a large number of external systems (e.g., 10,000)
- Postgres tuning and connection management as the number of clients grows
- Awaiting the release of the RDS proxy for Postgres to help manage connections

Figure 4.2: A Summary of Topic 3 Generated by Claude 3(Criteria Based)

Model	Issue	Cause	Solution	Rationale	Action	Done[1]
Claude 3 (Naïve Use)	-0.2	-0.4	-0.1	-0.37	-0.17	-0.03
Claude 3 (ADR Based)	-0.1	-0.4	-0.03	-0.23	-0.17	-0.13
Claude 3 (Criteria Based)	-0.2	-0.37	-0.17	-0.37	-0.07	-0.1
Claude 3 (Essence Based)	-0.13	-0.23	-0.13	-0.17	-0.2	0
GPT-4 (Naïve Use)	-0.2	-0.33	-0.13	-0.23	-0.1	-0.07
GPT-4 (ADR Based)	-0.17	-0.4	-0.03	-0.27	-0.17	-0.13
GPT-4 (Criteria Based)	-0.17	-0.43	-0.07	-0.37	-0.1	-0.1
GPT-4 (Essence Based)	-0.13	-0.2	-0.2	-0.17	-0.17	-0.03
Deepgram	-0.57	-0.7	-0.43	-0.6	-0.5	-0.13
Fireflies	-0.13	-0.37	-0.17	-0.23	-0.13	-0.07
Krisp	-0.13	-0.5	-0.2	-0.5	-0.37	-0.07
MeetGeek	-0.3	-0.6	-0.3	-0.57	-0.57	-0.13
Otter.ai	-0.17	-0.33	-0.3	-0.33	-0.47	-0.1
Sembly AI	-0.13	-0.5	-0.13	-0.37	-0.27	-0.13
Speechmatics	-0.07	-0.4	-0.07	-0.27	-0.4	-0.1
Zoom AI	-0.2	-0.47	-0.2	-0.33	-0.3	-0.13
LLMs	-0.16	-0.35	-0.11	-0.27	-0.14	-0.08
Meeting Tools	-0.21	-0.48	-0.23	-0.4	-0.38	-0.11
Overall	-0.19	-0.41	-0.17	-0.34	-0.26	-0.09

Table 4.3: The Mean Grades Lost Tools across the Six Questions

4.3 Length

As shown in Figure 4.4, the mean grade of all approaches is 3.34. Evident from Fireflies scoring 1.16 and Otter.ai scoring 5, however, is that the mean grade masks significant variability among the approaches.. Except Otter.ai, most approaches also exhibited a wide range of scores across the summaries each tested, with nearly all of them receiving both the highest (5) and lowest (0 or 1) grades across different summaries.

Meeting tools generally outperform LLMs. Compared to LLMs, meeting tools have an average score of 3.67, which is higher than the 3.02 average score of LLMs. To be more specific, six out of these eight meeting tools demonstrate better performance than any of the LLM approaches. The remaining two relatively poorer performers among the meeting tools are Fireflies, with a mean grade of 1.16 and Sembly AI (Mean = 3.24), which still performs better than some LLM approaches.

Model	Mean	Median	Mode	Max	Min	Counts(F)
Claude 3 (Naïve Use)	3.42	4	5	5	0	0
Claude 3 (ADR Based)	3.00	3	5	5	0	0
Claude 3 (Criteria Based)	3.48	4	5	5	1	1
Claude 3 (Essence Based)	3.47	4	5	5	0	0
GPT-4 (Naïve Use)	2.78	3	5	5	0	0
GPT-4 (ADR Based)	2.29	2	1	5	0	0
GPT-4 (Criteria Based)	2.76	3	5	5	0	0
GPT-4 (Essence Based)	2.93	3	5	5	0	0
Deepgram	4.60	5	5	5	1	0
Fireflies	1.16	1	0	4	0	0
Krisp	3.91	4	5	5	1	11
MeetGeek	4.14	4	5	5	2	16
Otter.ai	5.00	5	5	5	5	16
Sembly AI	3.24	4	4	5	1	4
Speechmatics	3.55	4	5	5	0	1
Zoom AI	3.73	4	4	5	0	0
LLMs	3.02					
Meeting Tools	3.67					
Overall	3.34					

Table 4.4: The Statistical Overview of Length Criteria Analysis

Otter.ai

22.PRIORITIES

Summary

•Prioritizing tasks based on business value. 0:00

oPrioritizing tasks based on business value requires a scoring mechanism to rank categories like scalability, performance, cost, and maintainability.

•Software issues and prioritization. 2:08

oUnknown Speaker discusses the importance of addressing nice-to-have features in software development.

Figure 4.3: A Summary of Topic 22 Generated by Otter.ai

Within meeting tools, Otter.ai demonstrated the best performance, obtaining a grade of 5 across all statistical measures. In contrast, Fireflies scored 0 in the most topics (mode = 0). As exemplified by the summary in Figure 4.3, Otter.ai provides a succinct summary annotated with timestamps for each discussion point, so the length of these summaries is therefore typically very short. Conversely, as shown Figure in 4.4, Fireflies typically

generates a single paragraph to summarize the entire meeting, followed by action items to list the tasks assigned during the meeting, an outline to capture the high-level structure of the discussion, and notes to document additional details and remarks. Therefore, although Fireflies captured more information, it also required many more words to articulate those additional contents, which led to its poorer performance in terms of length compared to other meeting tools. Regarding other meeting tools, Deepgram and MeetGeek performed commendably, with means exceeding 4 (4.60 and 4.18, respectively) and modes of 5.

Among the LLM approaches, the distribution of mean grades was observed to lie between 2.3 and 3.5. The Claude3 model, when engaged through prompts such as Naïve Use, Criteria-Based, and Essence-Based, demonstrated the best performance, with mean grades surpassing 3.4. Moreover, the median and mode for these approaches were at 4 and 5, respectively.

Within all prompts we explored, any approach powered by Claude3 outperforms those powered by GPT-4. To be more specific, the worst-performing Claude3 approach using the ADR Based prompt had a mean grade of 3.0, while the best-performing GPT-4 approach using the essence-based prompt only scored 2.93. Within the same model, the grades for the ADR Based prompts are significantly lower compared to other prompts. In the case of Claude 3, the mean grade for ADR Based prompts is 3, while for other prompts it exceeds 3.4. A similar trend is observed with GPT-4, where the ADR Based prompts receive comparatively lower grades than other prompts. Using GPT-4 as an example, Figure 4.5 and Figure 4.6 illustrate that, compared to summaries generated under Naïve-Use prompts, GPT-4 under ADR Based prompts captured more information pertaining to the decision-making process, such as Decision Drivers and Considered Options, in accordance with the definition of ADR (Architecture Decision Records). Unsurprisingly, recording such additional details consumed more words, leading to a lower grade.

Fireflies

AI meeting summary:

●The team discusses the definition of priority and how to determine it based on business value. They consider using a scoring mechanism to rank different categories and assign a scoring factor. They acknowledge the subjectivity of prioritization but believe that doing a few rankings can provide some objective insight. They also discuss the importance of addressing supportability issues and how smaller scope tasks can fit in with larger tickets. The conversation highlights the idea that nice-to-have features may become must-haves if they cause significant problems. Overall, they emphasize the purpose and flexibility of including nice-to-have items in their plans.

Action items:

●Based on the given transcript, it is difficult to identify specific follow-ups and action items. The conversation primarily focuses on discussing the definition of priority and how to determine the business value of different categories. While there are mentions of specific issues, such as supportability problems and slow local patient deletes, there is no clear indication of follow-ups or action items to address these issues.

●It is important to note that without further context or a more detailed discussion, it is challenging to determine specific follow-ups or action items from this transcript.

Outline:

●Chapter 1: Defining Priority and Business Value

●00:00 - 00:23: The discussion begins with a question about the definition of priority.

●00:11 - 00:15: The response suggests that business value is a determining factor for priority.

●00:23 - 00:56: The participants discuss a scoring mechanism to rank categories like scalability, performance, cost, maintainability, and supportability. The resulting score is considered as the business value.

●00:48 - 01:12: It is acknowledged that determining business value can be subjective, but it is necessary to avoid purely subjective descriptions.

●Chapter 2: Prioritization Process

●01:15 - 01:41: The participants discuss the importance of focusing on specific themes, such as supportability, to address current issues.

●01:41 - 02:04: The conversation shifts to the use of a six-grid quadrant to prioritize tasks.

●02:04 - 02:16: The participants emphasize the need to find work for everyone and explain how smaller scope tasks can fit in without hindering larger tickets.

●Chapter 3: Adjusting Priority Based on Impact

●02:16 - 02:57: The participants discuss a specific example where a "nice to have" task became a priority due to its impact on performance.

●02:57 - 03:12: They acknowledge the challenge of predicting when a small effort task becomes high value due to its impact on time and user experience.

●03:12 - 03:30: The participants highlight the importance of addressing issues promptly to avoid wasting time and customer complaints.

●Chapter 4: Nice to Have vs. Must Have

●03:30 - 03:46: The conversation concludes with the understanding that including "nice to have" tasks in the prioritization process serves a purpose and allows for potential inclusion in future releases.

●Please note that the chapter titles and content are inferred based on the transcript provided.

Notes:

●Definition of priority is based on business value.

●Prioritization involves ranking categories (such as scalability, performance, cost, maintainability, supportability) and assigning a scoring factor to determine business value.

●It is difficult to be completely objective in determining priority, as it involves subjective opinions.

●Prioritizing based on objective criteria can help eliminate subjective descriptions.

●Supportability issues may be prioritized if they are causing significant trouble in application support.

●Nice-to-have items, especially those in the small scope, are not thrown away as they can fit in with bigger tickets.

●Small scope items are considered to find work for everyone on the team.

●Some issues may be promoted to must-have status if they cause significant problems.

●It is challenging to predict whether a small effort will have high value, but fixing issues promptly can save time and prevent complaints.

●Nice-to-have items have a purpose and may be included in future iterations if not included in the current one.

Figure 4.4: A Summary of Topic 22 Generated by Fireflies

GPT-4 Config#1 - Naive Use

Meeting Summary - Elastic Logs Discussion

Elastic Log Configuration
- Discussion initiated by **Speaker 1** regarding the possibility of setting up elastic logs.
- **Speaker 2** expressed that it would be a nice addition but may not be a priority at this moment.

Implementation Considerations
- **Speaker 1** mentioned that setup involves configuration and a blue-green deployment.
- **Speaker 2** acknowledged that the setup process appears simple, likely involving terraform.

Prioritization and Scheduling
- The team categorized the setup of elastic logs as "nice to have small" items.
- **Speaker 2** suggested that these items could be included if there's extra time in a sprint.

Healthcare Cluster Commentary
- **Speaker 3** pointed out that the healthcare cluster might already have this enabled.
- The group agreed that it's worth looking into at a later time as a part of monitoring tasks.

Figure 4.5: A Summary of Topic 32 Generated by GPT-4 (Naïve-Use)

GPT-4 Config#2 - ADR-Like

Architecture Decision Record (ADR)

Decision: Setup of Elastic Logs

Status
Proposed

Context
The team discussed the prospect of setting up Elastic Logs for better monitoring and diagnostics. While it was acknowledged that this feature would be beneficial ("nice to have"), there was uncertainty about its priority in the current scope of work.

Decision Drivers
- The benefit of improved monitoring capabilities.
- The need to prioritize essential features over "nice to haves".
- The complexity and effort required to implement Elastic Logs.
- The availability of time within a sprint to accommodate such enhancements.

Considered Options
1. Implement Elastic Logs immediately as part of the current sprint.
2. Defer the Elastic Logs setup to a future sprint when there is available time.
3. Not implement Elastic Logs at all.

Decision Outcome
Chosen option: "2. Defer the Elastic Logs setup to a future sprint when there is available time.", because it allows the team to focus on more pressing tasks while maintaining the possibility of improving monitoring capabilities when the workload allows.

Consequences
- Elastic Logs will not be immediately available for the team, potentially postponing the benefits of improved monitoring.
- The team preserves the ability to prioritize critical tasks for the current sprint.
- Elastic Logs remain as a considered enhancement for future improvement if time permits.

Next Steps
- Assess the feasibility and complexity of implementing Elastic Logs.
- Add Elastic Logs to the backlog as a "nice to have" item to be picked up in future sprints if the schedule allows.

Figure 4.6: A Summary of Topic 32 Generated by GPT-4 (ADR Based)

To understand the impact brought by the original transcript, I analyzed the relationship between the grade and the length of the original transcript. As shown in Figure 4.7, the analysis reveals a strong correlation between the length grade and the length of the transcript. Specifically, the longer the original transcript, the higher the mean grade, indicating that the compression of the generated summary relative to the original text length is more pronounced for longer transcripts. Conversely, shorter meetings tend to result in lower mean grades, suggesting that summarization approaches struggle to achieve the same level of conciseness

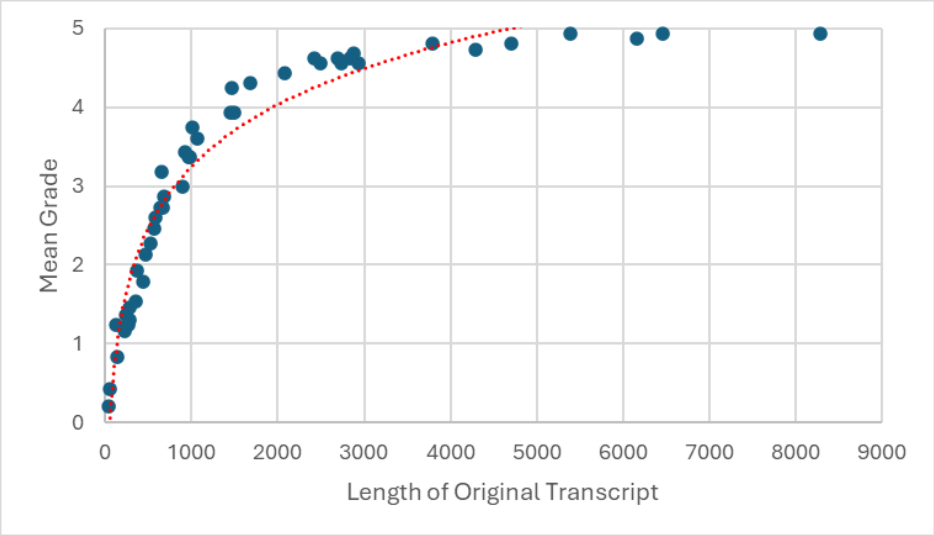

Figure 4.7: Relationship between Mean Grade and Length of Original Transcripts

when summarizing briefer meetings. A primary contributing factor to the lower grades is the tendency of some approaches to provide explanatory remarks when unable to capture certain information. Particularly in cases where a transcript is short, the transcript itself may lack comprehensive details such as action items and executed action items, but some approaches may still attempt to infer or elaborate on information not explicitly present in the meeting. As shown in Figure 4.8, for instance, the underlined words are entirely unrelated to the actual meeting content. Instead, these words are used to explain the reasons for missing information in the summary, yet they occupy a significant portion of the word count.

4.4 Structure

In general, as demonstrated in Table 4.5, the majority of meeting summary assistants evaluated received grades of 4 or 5 points, indicating a general trend towards high proficiency in generating structured summaries. According to the criteria introduced in Section 3.4.4, summaries with grade of 4 or 5 points contain at least multiple paragraphs, bullets, and headings.

Fireflies
AI meeting summary: ●The person in the transcript is sleeping at the beginning. They then discuss a non-critical task that complies with next gen policies and find it annoying. Someone mentions it's like a gut check. Action items: ●Based on the transcript, it is not explicitly clear what the follow-ups and action items are. The conversation seems to be more of a casual discussion rather than a specific task-oriented exchange. However, I can provide some inferences based on the context: ●1. Follow-up: Compliance to Next-Gen Policies: The speaker mentions that there is something related to compliance with next-generation policies that needs attention. It implies that there might be a need to review and ensure compliance with these policies. ●2. Action Item: Addressing Annoyances: The speaker mentions that something is "really annoying at large all the time." It suggests that there might be a need to identify and address the source of annoyance to mitigate its impact. ●3. Gut Check: The phrase "kind of like gut check" is used, which could indicate the need for a self-assessment or evaluation of a situation. However, it is not clear what specific action or follow-up is associated with this term. ●It is important to note that since the conversation is brief and lacks specific details, the mentioned inferences are based on general interpretations. Outline: ●Based on the provided transcript, it appears that there is not enough information to create an outline with chapters. The transcript consists of only three short sentences, which do not provide enough context or substance to identify distinct chapters or topics. Therefore, I am unable to create an outline with timestamps based on this limited information. Is there anything else I can assist you with? Notes: ●00:00: He's asleep. ●00:05: There's a task that is not critical but needs to be done to comply with next generation policies. It's simple but annoying. ●00:21: It's like a gut check.

Figure 4.8: A Summary of Topic 28 Generated by Fireflies

Model	Mean	Median	Mode	Max	Min	Counts(F)
Claude 3 (Naïve Use)	4.56	5	5	5	4	0
Claude 3 (ADR Based)	4.29	4	4	5	4	0
Claude 3 (Criteria Based)	4.66	5	5	5	4	1
Claude 3 (Essence Based)	3.91	4	4	4	4	0
GPT-4 (Naïve Use)	4.89	5	5	5	4	0
GPT-4 (ADR Based)	4.53	5	5	5	4	0
GPT-4 (Criteria Based)	4.96	5	5	5	4	0
GPT-4 (Essence Based)	4.02	4	4	5	4	0
Deepgram	0.00	0	0	0	0	0
Fireflies	5.00	5	5	5	5	0
Krisp	4.00	4	4	4	4	11
MeetGeek	5.00	5	5	5	5	16
Otter.ai	4.00	4	4	4	4	16
Sembly AI	5.00	5	5	5	5	4
Speechmatics	4.00	4	4	4	4	1
Zoom AI	3.84	4	4	4	3	0
LLMs	4.48					
Meeting Tools	3.86					
Overall	4.17					

Table 4.5: The Statistical Overview of Structure Criteria Analysis

In comparison to meeting tools, LLMs have a higher mean grade, with the former scoring an average of 3.86 and the latter 4.48. Additionally, the grades for LLMs are more variable. For instance, as shown in Figure 4.9 and Figure 4.10, with the same prompt, GPT-4 (Naïve Use) organized the summary according to different structures. The two summaries are of similar length, but a secondary heading was applied only in the summary of Topic 18 and not in the summary of Topic 19.

Among meeting tools, Fireflies stands out as the best option, having achieved a perfect grade across all evaluated topics. Additionally, it demonstrates the capability to generate summaries for transcripts of varying lengths. MeetGeek and Sembly AI also performed well, receiving the highest grades in all generated summaries. However, both were unable to produce summaries for shorter transcripts. In contrast, Deepgram received the lowest grade of 0 across all topics. Figure 4.11 demonstrates: why summaries generated by Deepgram consisted of disjointed sentences awkwardly concatenated without the use of conjunctions.

GPT-4 Config#1 - Naive Use

Meeting Summary

Support Issue Discussion

High-level Summary of Support Case
- **Issue in Testing**: It's not deemed high priority support, but involves a high priority client.
- **Current Problem**: Frequent errors from the API, possibly due to rate limiting from Cognito, and misunderstandings with AU Zero and user account configurations.

Action Items and Ownership
- **Volunteer**: Developer Y has volunteered to look into the issue.
- **Current Status**: No progress yet as team B have yet to start.
- **Recommendation**: Suggestion to let team B handle it since the issue emerged on the recently enabled test environment.

Important Insights and Updates
- **Authentication Issues**: Alerts about 500 errors and authentication issues have been observed.
- **Potential Solutions**: Might need a code change to implement caching and reduce the number of requests to Cognito.

Technical Discussion Regarding System SC
- **Account Creation**: Discussion about creating test user accounts on System SC test environment.
- **Access Controls**: No full control over the systems, and admins are responsible for creating accounts.
- **Meeting Setup**: A follow-up meeting is scheduled with the relevant team to address access and account creation issues.

Next Steps
- **Action for Speaker 5**: Request account creation for testing purpose or identify admin responsible for that task.
- **Subsequent Meeting**: Address inability to log in and reset passwords for certain users with the admin team in an upcoming meeting.

Figure 4.9: A Summary of Topic 18 Generated by GPT-4 (Naïve Use)

GPT-4 Config#1 - Naive Use

Date: Not provided
Attendees: Speaker 1, Speaker 2, Speaker 3
Key Points of Discussion:

System User Password for Local Testing
- Speaker 1 inquires where to obtain the system user password for a local environment for testing purposes.
- Speaker 2 suggests it may be the default password but doesn't recall the exact details.

Issues with Santa Cruz Environment
- Speaker 2 notes that System SC was experiencing issues which now seem to have been partly resolved, but there are still problems related to upgrades and access.
- Despite successful tests and adding new users, the precise cause of the persistent issues remained unidentified.

Problems with ESH Accessing API
- There was a discussion regarding ESH facing challenges in accessing the API.
- Speaker 1 is trying to troubleshoot this but is hindered due to uncertainties about obtaining the local system password.

Resolution Strategies
- Speaker 2 suggests either using a default password or using a function `GetUserSession()` to retrieve the system user session, which might contain the password.

Actions to be Taken
- Speaker 1 will look into the suggested methods to resolve the local password issue in order to proceed with fixing the issues in the environments.

Meeting participants may need to follow up on the resolution of environment issues and ensure access problems are resolved. Additional investigation may be required for the system user password retrieval method for local testing.

Figure 4.10: A Summary of Topic 19 Generated by GPT-4 (Naïve Use)

Furthermore, these summaries were devoid of any coherent structure, such as a central thesis supported by subsequent details, leading to poor readability. As a comparison, Figure 4.12 presents a 5-point summary generated by MeetGeek. This summary features clear multi-level headings and bullet points that effectively highlight the key points.

Deepgram

The speakers discuss the need for documentation on roles and the need for a support ticket for a B and Aws production access. They also discuss issues with traffic and a database server. They suggest creating a ticket for support and provide guidance on accessing the project's information. They discuss updating a list of names and setting up a bastion, and they briefly mention a training session and a problem with a Rca B team. They agree to create a ticket for a production admin and provide guidance on updating the list of names and removing an admin role.

Figure 4.11: A Summary of Topic 1 Generated by Deepgram

MeetGeek

Meeting Summary

In the meeting, there was a discussion about documentation and training for AWS. It was decided that to access the prod databases, one needs to go to RDS. A task was assigned to set up a bastion account and request it on the channel, with Developer B responsible for updating the list of names. Another topic discussed was creating support tickets for assistance, and a task was assigned to check with Developer S again for the PEM file. Lastly, there was a discussion about requesting a production admin role for Developer S, and it was mentioned that if the team doesn't have time to fix the cursor problem, the Team I will handle it tomorrow.

Next steps

- Set up a bastion account and request it on the HGH DevOps channel. Developer B needs to update the list of names.
- Check with Developer S again to see if he has the PEM file.

Highlights

1. Documentation and Training (4:2)

Discussion about documentation and training for AWS

- Action | To access the prod databases, you need to go to RDS and then you will be able to view all of them.

2. Requesting production admin role for Developer S (0:54)

Discussion about requesting production admin role for Developer S

- Fact | If the team doesn't have time to fix the cursor problem, the Irvine team will handle it tomorrow.

Figure 4.12: A Summary of Topic 1 Generated by MeetGeek

In the evaluation of the eight prompt configurations based on Claude 3 and GPT-4, the

Criteria-Based configuration demonstrated the highest performance, with the Naïve-Use configuration presenting a closely comparable performance. For Claude 3, the respective grades were 4.66 and 4.56, while for GPT-4, they were 4.93 and 4.76 respectively. Regarding the ADR-based prompts, Claude3 achieved a grade of 4 (mean 4.29) across most Topics, while GPT-4 performed slightly better with a mean grade of 4.67. By the definition of ADR, all ADR-based summaries were structured into paragraphs with headings such as Status, Context, and Decision. Moreover, upon observing all generated summaries, we found that LLMs always organize content in a bulleted format. Consequently, based on the criteria, the minimum grade would not fall below 4 (encompassing headings, bullets, and multiple paragraphs). However, whether a grade of 5 could be attained largely depended on whether LLMs provided secondary headings for each Decision. Lastly, for the Essence Based prompts, as the prompts consisted of six specific questions, this may have constrained the LLMs from further organizing the prompts into multiple headings and bullets under each question. Figure 4.13 illustrates why summaries generated by essence-based LLM approaches scored 4 in all topics. The sample summary consists of the question and answers organized by bullets, so it will minimally score 4 for including headings, bullets and multiple paragraphs by default.

4.5 Clarity

As illustrated in Table 4.6, all evaluated approaches exhibited remarkable proficiency in achieving Clarity, with consistently high scores across all topics. Reading obstacles rarely appeared in the generated summaries, with only one instance occurring in summaries produced by Fireflies and GPT-4 Criteria. This indicates that the generated summaries have achieved a high standard in terms of textual unambiguity and clear expression, aligning with previous research findings [2].

GPT-4 Config#4 Essence Based
1. **Issue/Problem Being Discussed:** - Whether to set up elastic logs. 2. **Why It Happens/Causes:** - The discussion implies it's a "nice to have" feature, not a necessity. The decision is based on its perceived utility versus effort and prioritization among other tasks. 3. **Solution/Resolution:** - The solution appears to be to categorize the setup of elastic logs as a "nice to have" task that can be pulled into a sprint when there is extra time, after more critical tasks are addressed. 4. **Why This Solution/Resolution:** - This solution is chosen due to its simplicity to implement (mentioned as being manageable with Terraform and configuration) and its non-critical nature. It's a low-priority enhancement that can improve monitoring but doesn't warrant immediate action over other tasks. 5. **Concrete Action Items:** - Identify and classify similar "nice to have" tasks. - Prioritize critical tasks for upcoming sprints. - Explore how healthcare has enabled it as a potential reference. - Allocate time in future sprints for these lower-priority enhancements if applicable. 6. **Action Executed During the Meeting:** - Agreed to categorize the setup of elastic logs as a non-critical, "nice to have" task. - Identified healthcare's implementation of a similar feature as a potential reference for future action.

Figure 4.13: A Summary of Topic 32 Generated by GPT-4 (Essence Based)

Model	Mean	Median	Mode	Max	Min	Counts(F)
Claude 3 (Naïve Use)	5.00	5	5	5	5	0
Claude 3 (ADR Based)	5.00	5	5	5	5	0
Claude 3 (Criteria Based)	5.00	5	5	5	5	1
Claude 3 (Essence Based)	5.00	5	5	5	5	0
GPT-4 (Naïve Use)	5.00	5	5	5	4	0
GPT-4 (ADR Based)	5.00	5	5	5	5	0
GPT-4 (Criteria Based)	4.98	5	5	5	5	0
GPT-4 (Essence Based)	5.00	5	5	5	5	0
Deepgram	5.00	5	5	5	4	0
Fireflies	4.98	5	5	5	5	0
Krisp	5.00	5	5	5	5	11
MeetGeek	5.00	5	5	5	5	16
Otter.ai	5.00	5	5	5	4	16
Sembly AI	5.00	5	5	5	5	4
Speechmatics	5.00	5	5	5	5	1
Zoom AI	5.00	5	5	5	5	0
LLMs	5.00					
Meeting Tools	5.00					
Overall	5.00					

Table 4.6: The Statistical Overview of Clarity Criteria Analysis

Taking the reading obstacle in Fireflies as an example, one bullet point states:

> "-No significant issues identified with the current workflows impacting other clients."

It could be interpreted in two ways: either no significant issues were found at all, or issues were found but they were not deemed significant. Additionally, it's unclear whether the phrase "impacting other clients" modifies "workflows" or "issues." This lack of clarity could lead to misunderstandings about the current state of the current workflows and their potential impact on clients.

4.6 Overall Analysis

Figure 4.14 provides a comprehensive performance evaluation by accumulating all scores per approach. The data reveals notable differences in performance across the evaluated approaches, with total grades ranging from 16.31 to 21.90 out of a maximum possible score of 25. The highest-scoring approach is Claude-3 (Criteria Based), which achieves a total score of 21.90, indicating balanced good performance across all five metrics. Conversely, Deepgram has the lowest total score of 16.31, primarily due to its poor performance in the Structure metric, where it scores 0. Across the five metrics, truthfulness and clarity emerge as the strongest areas, with most approaches consistently scoring at or near the maximum of 5. This suggests that almost all tools are generally reliable in providing accurate and clear outputs. However, the metrics of length and essence exhibit more variability and appear to be the weakest areas for many approaches, indicating challenges in maintaining concise yet meaningful content.

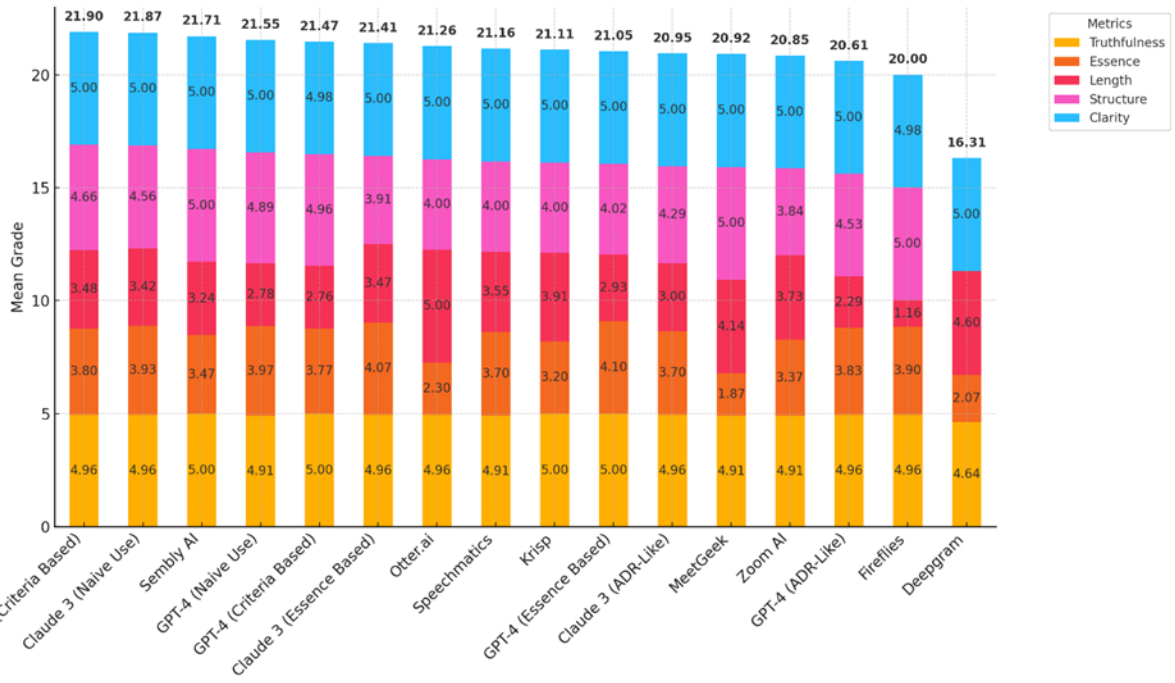

Figure 4.14: Overall Performance Comparison

There are notable differences in performance between LLM approaches and meeting tools. LLM approaches tend to have more uniform scores across all five metrics, while meeting tools show more variation across metrics. Although meeting tools often perform as well as LLM approaches in truthfulness and clarity, their scores in essence and structure are less consistent. For instance, compared to its performance in truthfulness and clarity, Deepgram shows significant weakness in capturing essential information and organizing the summary in a clear structure. This uneven performance suggests that, while meeting tools are capable in specific tasks, they struggle with providing the same level of overall quality and consistency as LLM approaches. In contrast, LLM approaches exhibit relatively more balanced performance, making them more reliable for tasks requiring comprehensive language understanding and content generation.

Among the LLM approaches, there is a sizeable variation in performance across different configurations, though all maintain relatively high overall scores. The Claude-3 (Criteria Based) stands out with the highest total score of 21.90, indicating its strong performance, particularly in essence and structure, where it balances the preservation of meaning with well-organized content. Other configurations, such as the Claude-3 (Naïve Use) also perform well but tend to score slightly lower in length and structure.

Notably, the ADR Based configurations, whether in Claude-3 or GPT-4, consistently underperform compared to other configurations, with scores generally lower across most metrics. Although ADR is widely used in recording decision making, ADR Based approaches do not exhibit any strengths and particularly fall short in essence and length. More interestingly, despite the assumption that more detailed prompts tailored to specific criteria might yield better results, in both LLMs, the Naïve Use configurations perform similarly to the more sophisticated Criteria Based prompts. For instance, GPT-4 (Naïve Use) scores higher in total (21.55) compared to Claude-3 (Criteria Based) (21.47), even though the latter is explicitly designed to align with the five metrics. This suggests that specific or complex instructions

may not allow the LLMs to leverage their inherent strengths in summarization compared to simpler, more straightforward prompts

Within meeting tools, while all tools generally perform well in truthfulness and clarity, some have notable weaknesses in one or two other metrics. Sembly AI and Otter.ai stand out for their strong overall performance, particularly excelling in structure, but also showing some weaknesses in capturing essential information. Conversely, Deepgram is the weakest performer, particularly struggling with structure and essence where it scores 0 and 2.07 respectively. Although it performs reasonably in length, the lack of structural coherence and key information severely limits its capability. In comparison, Fireflies excels in capturing the essence of the meeting, but significantly underperforms in the length metric, scoring only 1.16, which suggests it struggles to maintain an appropriate level of conciseness.

As shown in Figure 4.15, if we focus only on truthfulness and essence, the rankings of the approaches change notably. GPT-4 (Essence Based) emerges as the leader, reflecting its strong emphasis on these two metrics, closely followed by Claude 3 (Essence Based), which also ranks highly. These two approaches, which might not have been the top performers in the overall evaluation, show their potential when only these core metrics are considered. Outstanding performance of essence-based prompt further demonstrates the strength of LLMs when provided with clear and specific instructions.

Claude 3 (Naive Use) and GPT-4 (Naïve Use) maintain strong positions, but others such as Otter.ai and MeetGeek drop significantly in the rankings, highlighting their relative weakness in truthfulness and essence.

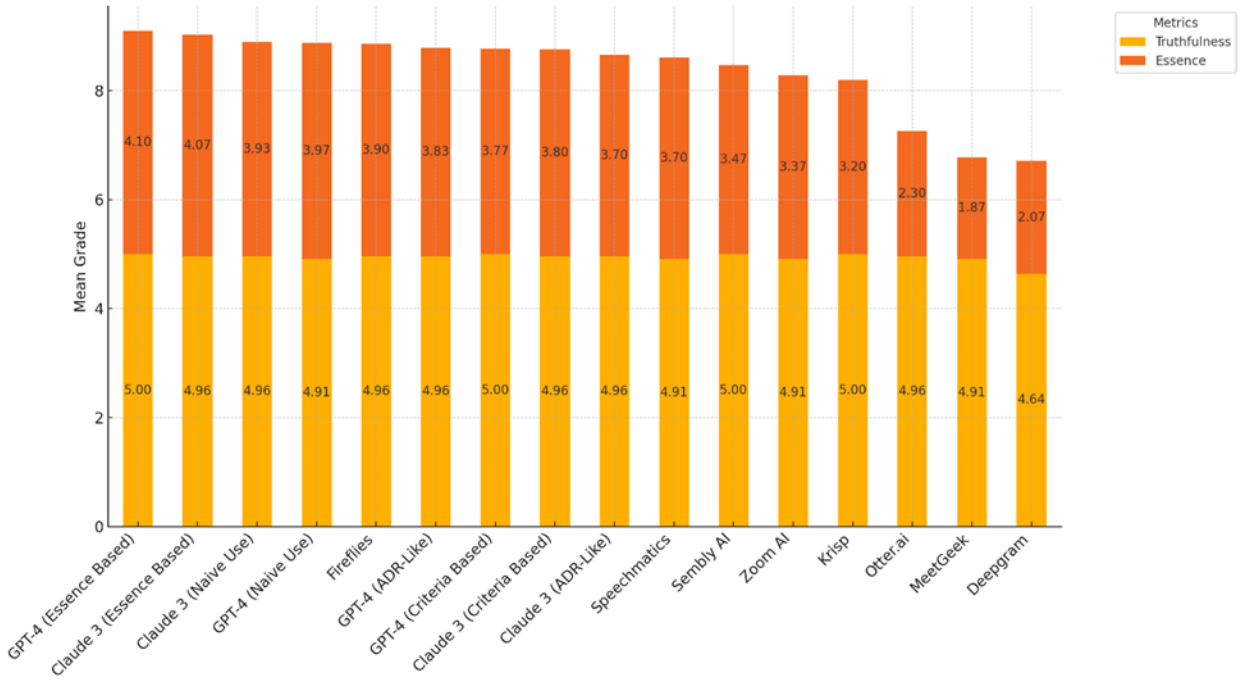

Figure 4.15: Comparison of Truthfulness and Essence Performance

Chapter 5

Discussion

The results of this comprehensive evaluation of meeting summary generation approaches reveal several insights and implications.

5.1 Absence of a Universal Best Approach

A significant finding from this study is that no single approach excels across all metrics. This observation underscores the complexity of meeting summarization, and the inherent trade-offs involved in optimizing for different aspects of summary quality.

For instance, while Fireflies demonstrated superior performance in capturing the essence of meetings, it struggled significantly with length management, scoring only 1.16 in this metric. This suggests that its strength in comprehensive information capture comes at the cost of conciseness. Conversely, Otter.ai excelled in producing concise summaries, achieving a perfect score of 5 in the length metric, but it showed weaknesses in capturing essential information, with a relatively low score of 2.30 in the essence metric.

Similarly, among LLM approaches, we observe varying strengths across different configurations. The Claude-3 (Essence Based) configuration achieved the highest overall score, particularly excelling in essence capture and structural organization. However, it did not outperform other configurations in all metrics, notably scoring lower in the length metric compared to some other approaches.

5.2 Trade-offs among Essence, Length, and Structure

Another finding from this study is the evident trade-off among length essence and structure. As depicted in Figure 5.1, configurations such as Otter.ai and Deepgram, positioned on the far right of the horizonal axis, show high length scores but fall significantly below the median line for the average of essence and structure. Specifically, Otter.ai, which achieves the maximum length score of 5, has a markedly lower average score for essence and structure, placing it well below the red median threshold line at 4.03. This suggests that while Otter.ai excels in generating detailed or lengthy content, it sacrifices the ability to capture essential information or maintain coherent structure. Similarly, Deepgram, despite its high length grade, shows a critical deficiency in Structure, as indicated by its zero score in that category, highlighting the trade-off between producing extensive content and maintaining content quality.

This trade-off can be attributed to the inherent challenges in balancing different aspects of summary quality. To improve essence capture, approaches may tend to include more information from the original meeting, inevitably increasing the summary's length. This is evident in cases like Fireflies, which achieves a high essence score but struggles with conciseness. Conversely, to enhance structure, approaches often employ techniques such as bullet points, headings, and subheadings. While these structural elements improve readability and organization, they can also contribute to increased length. For instance, Fireflies and GPT-4

(ADR Based) both achieve perfect structure scores but at the cost of lower length scores, illustrating how the inclusion of structural elements can impact summary brevity.

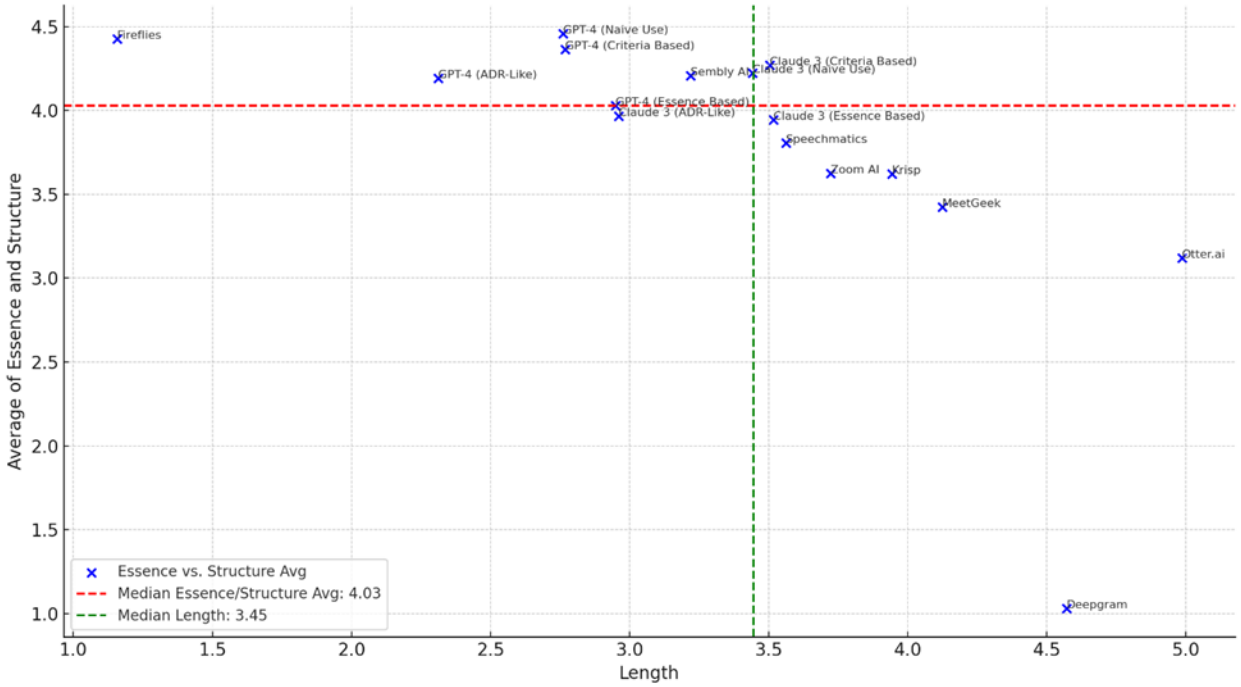

Figure 5.1: Trade-off between Length and Essence/Structure

5.3 Independence of Truthfulness and Clarity from Length

An intriguing finding from this study is the apparent lack of significant correlation between the Length metric and the metrics of Truthfulness and Clarity. This observation challenges the intuitive assumption that longer summaries might be more prone to errors or lack clarity.

As shown Figure 5.2, the lines representing truthfulness (orange) and clarity (red) are almost perfectly flat, hovering around the maximum score of 5 for most configurations. This consistency indicates that these two metrics remain stable and high across all models, irrespective of the variations in length. The fact that these lines do not dip or rise significantly suggests that Truthfulness and Clarity are largely unaffected by the content's length. Even as the

length metric varies significantly—ranging from as low as 1.16 for Fireflies to as high as 5 for Otter.ai—the truthfulness and clarity scores stay nearly constant.

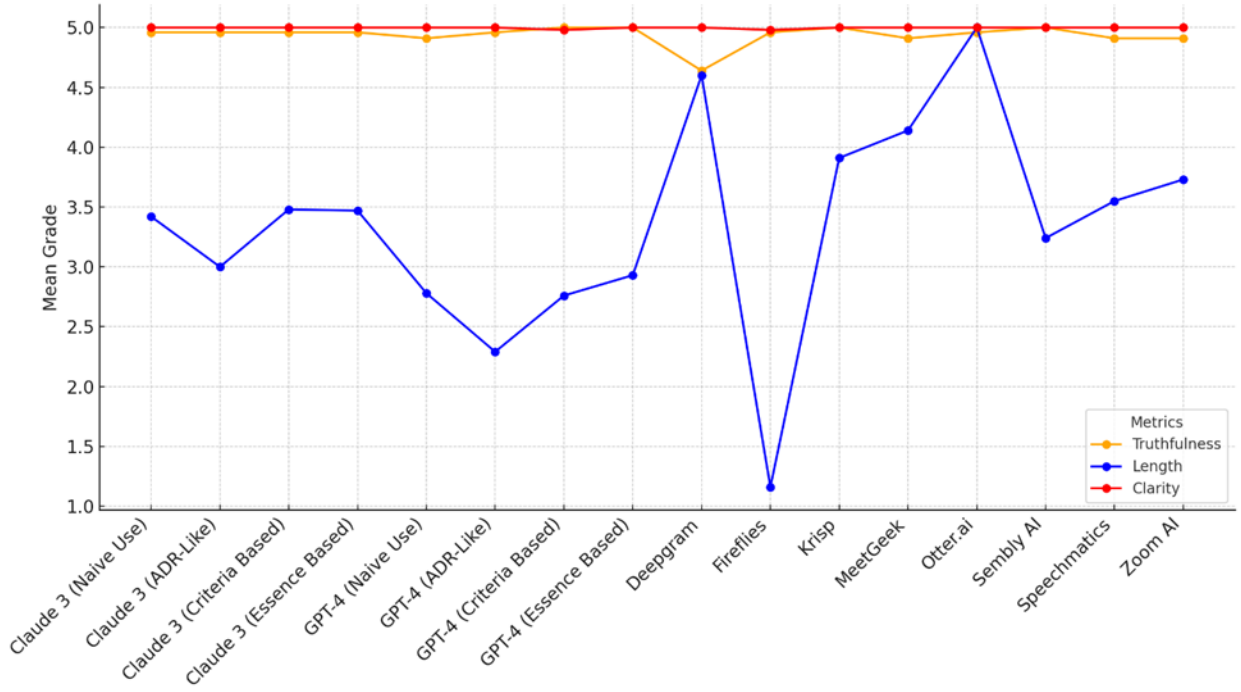

Figure 5.2: Independence of Truthfulness and Clarity from Length

5.4 New Trends in Meeting Tools

The landscape of meeting tools is rapidly evolving, with a notable trend towards the integration of interactive AI capabilities, similar to those found in ChatGPT and Claude 3. This development represents a significant shift in how developers can interact with and retrieve information from meeting summaries. Chat-based AI systems offer a level of interactivity that traditional static summaries cannot match, allowing users to ask follow-up questions, request clarifications, or seek additional context about specific points discussed in a meeting. For software developers, this interactivity can be invaluable when reviewing meeting content, enabling them to directly query the AI about particular decisions, action items, or technical discussions, and receive targeted responses that save time and enhance understanding.

Our findings in Section 4.2 highlight a crucial advantage of interactive AI systems: their ability to provide more accurate and relevant information when given specific prompts. The essence-based prompts consistently outperformed other approaches in both GPT-4 and Claude 3 models, demonstrating that these systems are highly capable of extracting and presenting targeted information when explicitly instructed in the desired forms of the response. This capability aligns well with the diverse information needs in software development contexts, where different team members might need different aspects of a meeting summarized.

The integration of chat-based AI into meeting tools is an emerging trend that promises to revolutionize how teams interact with meeting content. For example, Fireflies has already incorporated a chat feature into its platform, allowing users to interact with meeting summaries in a more dynamic way. This integration may represent a potential future direction for meeting tools across the board, offering more flexible information retrieval, contextual understanding, personalized summaries, and enhanced follow-up capabilities.

Chapter 6

Limitations and Threats to Validity

This chapter examines the limitations and threats to validity in our study of meeting summarization approaches for software development. I address constraints in the research scope, potential biases in our evaluation process, and challenges to the generalizability of our findings.

6.1 Limitations

A significant limitation of this study is its focus solely on software maintenance meetings. While these meetings are an important aspect of software development, they represent only one type of the diverse range of meetings that occur throughout the software development lifecycle. Other types of meetings, such as requirements gathering, design discussions, sprint planning, or project retrospectives, may have different characteristics and information structures. Consequently, the performance of summarization approaches evaluated in this study may not necessarily generalize to these other meeting types. The unique vocabulary, discussion patterns, and decision-making processes in different types of software development

meetings could potentially lead to varied summarization challenges and outcomes.

Another limitation of this study is the constrained scope of meeting tools and LLLMs evaluated. The rapidly evolving landscape of AI and meeting summarization technologies means that new tools and models are continuously being developed and released. The approaches examined in this study represent only a snapshot of the available technologies at the time of the research. As a result, the findings of this study may not fully reflect the current state-of-the-art in meeting summarization capabilities.

6.2 Threats to Validity

Internal Validity: For truthfulness and clarity, the assessment was conducted by a single grader, which introduces the potential for individual bias in the evaluation. The subjective nature of these criteria means that different graders might interpret and score the summaries differently. Similarly, for the essence criterion, only one person was responsible for creating the reference summary, with another single grader evaluating the generated summaries against this reference. This limited perspective could lead to a narrow interpretation of what constitutes the essential content of a meeting.

External Validity: All the meetings were sourced from a single team working on one specific project within a single team. This narrow focus potentially limits the generalizability of our findings to broader software development contexts. Different organizations, teams, and projects may have varying meeting dynamics, communication styles, technical vocabularies, and discussion complexities. These factors could significantly influence the performance of summarization approaches. Furthermore, the segmentation of meetings into topics resulted in some exceptionally short segments, which may not accurately reflect the complexity and length of real-world software development meetings.

Construct Threats: In the evaluation of comprehensive performance, equal weights are assigned to the five metrics (Truthfulness, Essence, Length, Structure, and Clarity). This uniform weighting assumes that each metric contributes equally to the quality and effectiveness of a meeting summary. However, in real-world software development contexts, the relative importance of these metrics may vary depending on the specific needs of the team, the nature of the project, or the purpose of the meeting. This simplification could potentially lead to conclusions about overall performance that do not align with the practical value of summaries in diverse professional settings.

Chapter 7

Conclusion

This thesis provides a comprehensive evaluation of 16 meeting summarization approaches across 45 software maintenance meeting topics. The findings highlight that no single approach excels across all metrics, underscoring the complexity of meeting summarization and the inherent trade-offs involved. While most approaches demonstrated high performance in truthfulness and clarity, significant variability was observed in essence, length, and structure, with LLMs generally showing more consistent performance across metrics compared to meeting tools. The study also revealed the impact of prompt design on LLM performance and the emergence of interactive AI as a promising trend in meeting tools. These findings provide a snapshot of current summarization technologies' strengths and limitations, which may help inform tool selection and usage in software development teams and provide insight for the next-generation summarization tools.

Chapter 8

Future Work

Building on this study's findings and addressing its limitations, future research will focus on three key areas:

Long-term Impact Assessment: Conduct longitudinal studies to evaluate the long-term effects of using automated summarization tools on team communication, decision-making, and overall project outcomes in software development contexts. This could include assessing how the use of these tools impacts knowledge retention, task completion rates, and project success over time.

Multi-grader Evaluation: Implement a multi-rater evaluation system to reduce individual bias and provide a more comprehensive assessment of summary quality. This could involve software developers, project managers, and other stakeholders rating summaries based on their specific needs and perspectives.

Cross-domain Comparison: Expand the evaluation to include different types of software development meetings (e.g., sprint planning, code reviews, retrospectives) and compare how summarization approaches perform across these various contexts. This would provide in-

sights into the versatility and limitations of current summarization technologies.

Bibliography

[1] S. W. Ambler. Agile modeling: A brief overview. `https://www.agilemodeling.com/`, 2001.

[2] A. Bhaskar, A. R. Fabbri, and G. Durrett. Prompted opinion summarization with gpt-3.5. *arXiv preprint arXiv:2211.15914*, 2022.

[3] H. T. Dang. Overview of duc 2005 (draft). In *Proc. Document Understanding Conferences*, 2005. as cited in E. Lloret, L. Plaza, and A. Aker, "The challenging task of summary evaluation: an overview," Language Resources and Evaluation, vol. 52, pp. 101-148, 2018.

[4] K. Dikert, M. Paasivaara, and C. Lassenius. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 119:87–108, 2016.

[5] A. R. Fabbri, W. Kryściński, B. McCann, C. Xiong, R. Socher, and D. Radev. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409, 2021.

[6] A. Fiori. *Innovative document summarization techniques: Revolutionizing knowledge understanding: Revolutionizing knowledge understanding*. IGI Global, 2014.

[7] Fireflies.ai. Homepage. `https://Fireflies.ai/`. Accessed: Jun. 4, 2024.

[8] G2. G2 - discover the best business software and services. `https://www.g2.com/`. Accessed on: Jun. 25, 2024.

[9] T. Goyal, J. J. Li, and G. Durrett. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*, 2022.

[10] J. Harjamäki et al. The report of 85 ai tools: Genai content production: Enhancing repeatability and automation with chatgpt, 2024.

[11] A. A. Keshlaf, A. A. Alahresh, and M. K. Aswad. Factors influencing the use of online meeting tools. In *2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA*, pages 908–912, Tripoli, Libya, 2021.

[12] J. Klünder, K. Schneider, F. Kortum, J. Straube, L. Handke, and S. Kauffeld. Communication in teams-an expression of social conflicts. In *Human-Centered and Error-Resilient Systems Development: IFIP WG 13.2/13.5 Joint Working Conference, 6th International Conference on Human-Centered Software Engineering, HCSE 2016, and 8th International Conference on Human Error, Safety, and System Development, HESSD 2016, Stockholm, Sweden, August 29-31, 2016, Proceedings 8*, pages 111–129. Springer, 2016.

[13] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004. as cited in D. Deutsch, R. Dror, and D. Roth, "A Statistical Analysis of Summarization Evaluation Metrics Using Resampling Methods," Trans. Assoc. Comput. Linguistics, vol. 9, pp. 1132-1146, 2021.

[14] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[15] E. Lloret. The challenging task of summary evaluation: an overview. *Lang. Resour. Eval.*, 52(1):101–148, 2018.

[16] I. Mani. *Automatic summarization*, volume 3. John Benjamins Publishing Company, Amsterdam, 2001. as cited in E. Lloret, L. Plaza, and A. Aker, "The challenging task of summary evaluation: an overview," Language Resources and Evaluation, vol. 52, pp. 101-148, 2018.

[17] MeetGeek. Homepage. `https://meetgeek.ai/`. Accessed: Jun. 11, 2024.

[18] A. Nenkova, R. Passonneau, and K. McKeown. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4(2):4–es, 2007.

[19] Otter.ai. Homepage. `https://otter.ai/`. Accessed: Jun. 4, 2024.

[20] K. Schneider, O. Liskin, H. Paulsen, and S. Kauffeld. Media, mood, and meetings: Related to project success? *ACM Transactions on Computing Education (TOCE)*, 15(4):1–33, 2015.

[21] H. Shakil, A. M. Mahi, P. Nguyen, Z. Ortiz, and M. T. Mardini. Evaluating text summaries generated by large language models using openai's gpt. *arXiv preprint arXiv:2405.04053*, 2024.

[22] A. M. Soria, T. Lopez, E. Seero, N. Mashhadi, E. Evans, J. Burge, and A. Van der Hoek. Characterizing software maintenance meetings: Information shared, discussion outcomes, and information captured. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13, 2024.

[23] A. M. Soria, A. van der Hoek, and J. Burge. Recurring distributed software maintenance meetings: toward an initial understanding. In *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering*, pages 21–25, 2022.

[24] V. Stray and N. B. Moe. Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and slack. *Journal of Systems and Software*, 170:110717, 2020.

[25] M. Taş and A. Kiraz. A model for the acceptance and use of online meeting tools. *Systems*, 11(12):558, 2023.

[26] S. Teufel and H. van Halteren. Evaluating information content by factoid analysis: Human annotation and stability. In *Proc. Conf. Empirical Methods in Natural Language Processing*, pages 419–426, 2004. as cited in E. Lloret, L. Plaza, and A. Aker, "The challenging task of summary evaluation: an overview," Language Resources and Evaluation, vol. 52, pp. 101-148, 2018.

[27] E. Voorhees. Overview of the trec 2003 question answering track. In *Proc. Twelfth Text Retrieval Conf. (TREC)*, 2003. as cited in E. Lloret, L. Plaza, and A. Aker, "The challenging task of summary evaluation: an overview," Language Resources and Evaluation, vol. 52, pp. 101-148, 2018.

[28] S. Whittaker, R. Laban, and S. Tucker. Analysing meeting records: An ethnographic study and technological implications. In *Machine Learning for Multimodal Interaction: Second International Workshop, MLMI 2005, Edinburgh, UK, July 11-13, 2005, Revised Selected Papers*, volume 2. Springer Berlin Heidelberg, 2006.

[29] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. McKeown, and T. B. Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024.

[30] Y. Zhou, F. Ringeval, and F. Portet. Can gpt models follow human summarization guidelines? evaluating chatgpt and gpt-4 for dialogue summarization. *arXiv preprint arXiv:2310.16810*, 2023.