2025 年度　卒業論文

LLM による打合せログからの要求仕様書生成

公立諏訪東京理科大学
工学部　情報応用工学科

T121009 飯田 琉斗
指導教員　山田　哲靖

目次

1. はじめに

1.1. 研究背景

現代のソフトウェア・アプリケーション開発において、議事録から要求仕様書を作成する作業は主に人手に依存しており、以下のような課題に直面している。
【コスト・スピードの問題】
議事録から要件書を作成するのに、数時間を要し、PM らの作業時間がそこに割かれている。また、重要なタスクを優先するため、議事録の整理が後回しになり、会議内容の鮮度が落ちる問題が生じている。
特にログの量が増大するにつれ、簡略化や省略がしばしば発生する傾向がある。

【主観バイアスと選択的要約の問題】
最も深刻な問題は、作成者の理解に引っ張られた「都合の良い整理」である。
作成者の「重要そう」に見える部分を詳しく記載する一方で、致命的な制約や例外条件を見落とすケースが頻発している。
さらに、組織の上下関係により、一部の発言が軽視されるなど発言バイアスも問題となっている。

【形式・粒度の不一致】
同じテンプレートを使用しても、作成者によって記載内容や粒度が微妙に異なり、チーム内での情報共有に支障をきたしている。
特に、担当者の引継ぎ時には、前任者の書き方の癖が残り、一貫性のない文書が蓄積される問題がある。

【単純ミスの多発】
コピー&ペーストのミス、日時や数字の打ち間違い、別トピックとの混同といったヒューマンエラーが避けられず、これらが後の開発工程での混乱や手戻りの原因となる。
これらの課題を受けて、近年注目されている生成 AI を活用できないかと考えた。
本研究では、OpenAI が開発した“GPT-4o”を使用する。GPT-4o は高度な自然言語理解能力を持ち、テキストの文脈理解や要約処理において優れた性能を示しており、議事録処理への応用に適していると考えられる。

1.2. 研究目的

前述した GPT-4o をそのまま使うだけでは、人による課題が解決されず、出力の不安定性や情報の正確性に関する課題が明らかになった。

これらの人手と AI 単純適用の両方の問題を解決するため、本研究では、以下を目標とする:

1. 事実・推測の混同防止
2. 重要項目の欠落防止
3. 時系列の正確な保持
4. 推測補完の除去
5. 書式形式の統一
6. 処理コストの大幅削減
7. ヒューマンエラーの排除

生成 AI の利点を活かしつつ、その問題を技術的に補完することで、実用的な議事録自動処理システムを開発し、その有効性を評価することを目標とする。

2. 生成 AI の可能性と限界

2.1. 生成 AI に期待される役割

1 章で示した人手による議事録処理の課題に対し、生成 AI は以下の解決可能性を持つ。

【コスト・スピード問題の解決】

GPT-4o は、大量のテキストを高速で処理する能力を持ち、数時間を要する人で作業を短時間で完了できる可能性がある。また、重要タスクの優先により後回しになっていた整理タスクを、会議直後に即座に実行できることで、会議内容の鮮度を保った状態での文書化が期待される。

【主観バイアスの排除】

AI は作成者の理解や立場に影響されない客観的な情報抽出が可能である。組織の上下関係や個人の主観に左右されず、すべての発言を平等に分析することで、「都合の良い整理」や「発言バイアス」を排除できる可能性がある。

【形式・粒度の統一】

生成 AI は一貫した基準で情報を処理するため、作成者による記載内容や粒度のばらつきを解消できる。また、担当者の引き継ぎ時にも、同一の処理ロジックにより一貫した品質の文書を生成することが期待される。

【ヒューマンエラーの排除】

コピー&ペーストのミス、日時や数字の打ち間違い、別トピックとの混同といった単純ミスをゼロにできる可能性がある。

【高度な自然言語理解】
GPT-4o は文脈理解、要約、情報抽出において高い性能を示しており、人間の発言の意図を理解し、重要な決定事項を的確に識別する能力が期待される。
これらの期待に基づき、本研究では、GPT-4o の議事録処理への適用可能性を実験検証することとした。

2.2. 実験: 素の生成 AI での生成処理

【実験設定】

使用モデル	GPT-4o
入力データ	Web 開発の打ち合わせログ
処理方法	JSON ファイルをインポート
プロンプト	提示した議事録から開発チーム向けの要件定義書を作成してください。
実験回数	同一データ・プロンプトで 10 回実行
評価項目	決定事項の抽出率、文書構造、処理時間

【実験データ】

プロジェクト	Web アプリケーション開発
データ形式	JSON
参画者	4 名（PM, クライアント, エンジニア, デザイナー）
議論内容	ログイン機能の仕様、UI 設計、通知機能の取り扱い、など
雑談等のノイズ混在	あり

ファイル内容（一部抜粋):

```
[
  {
    "speaker": "Client",
    "text": "ログイン画面はアプリ起動直後に表示されるようにしたいです。",
    "timestamp": "2024-11-20T13:00:00"
  },
```

```
{
    "speaker": "Engineer",
    "text": "起動直後のログイン表示、技術的には可能です。",
    "timestamp": "2024-11-20T13:01:00"
},
{
    "speaker": "PM",
    "text": "では、ログイン初期表示は一旦の方針ということで。",
    "timestamp": "2024-11-20T13:02:00"
},
```


【実験手順】

1. 実案件をイメージしたサンプル議事録を JSON 形式で準備
2. 作成した JSON ファイルとプロンプトを GPT-4o に入力
3. 出力結果を記録
4. 1.~3.を 10 回行い、出力結果を比較

【実際の決定事項】

人手による分析では、以下の 9 つの決定事項が確認された:

1. ログイン画面をアプリ起動直後に表示
2. 通知機能を今回リリースから除外
3. 管理者に編集・削除権限を付与
4. 初回表示を 3 秒以内の非機能要件として設定
5. エラーメッセージ文言を「メールまたはパスワードが違います」に設定
6. ログ保持期間を 90 日に設定

7. 簡易チュートリアルを今回の範囲に追加
8. ログインボタンの位置を中央下寄せに変更
9. リリース候補日を 12 月 20 日に暫定

通知機能について「除外→簡易表示を検討→最終的に除外」という議論の変遷がある。期待される結果は、AI が文脈を理解して最終決定を正しく抽出でき、全 9 項目が正しく抽出できていることである。

2.3. 観測された問題

前節にて、生成 AI への期待とは大きく下回り、いくつかの問題点が明らかになった。

【具体的な事例】

《第 3 回　出力》

```
# アプリケーション要件定義書



## 決定事項

- ログイン画面をアプリ起動時に表示する

- 管理者権限でユーザーの編集・削除を可能にする

- リリース予定日を 12 月 20 日にする



## 検討事項

- 通知機能については継続検討

- ボタンレイアウトは複数案で比較予定
```

図 1 素の AI 実験 第 3 回出力

《第 7 回　出力》

要件定義書

決定事項

- ログイン画面をアプリ起動時に表示
- 管理者権限でユーザーの編集・削除を可能にする
- 通知機能は、今回のリリースから除外

追加機能

- 3 ステップの簡易チュートリアルを実装
- ログインボタンを中央下寄席で配置

その他

色彩設計やデザイン詳細は次回の会議で決定予定

図 2 素の AI 実験 第 7 回出力

《第 9 回　出力》

| カテゴリ | 項目 | 決定内容 | 備考 |

|----------|------|----------|------|

| UI | ログイン表示 | 起動時表示 | 確定 |

| 機能 | 通知 | 除外 | 今回リリース外

| 権限 | 管理者 | 編集・削除可 | 正式仕様 |

図 3 素の AI 実験 第 9 回出力

2.3.1. 出力のばらつき

【問題の概要・影響】
同一の議事録とプロンプトを用いた命令を行ったとしても、常に同じ結果が出力されることはなかった。抽出数が全項目数に達していないといったばらつきが見られた。また、生成 AI の予測・推測能力によって項目内容の拡張が見られることもある。

【分析】
以下、表 1 は、当実験を 10 回行った測定の統計である。

表 1 素の AI 実験結果

最小抽出数	5 項目
最大抽出数	8 項目
平均抽出数	6.2 項目
実際の項目数	9 項目
平均抽出率	68.89%

2.3.2. 文脈情報の欠落

【問題の概要・影響】
重要な機能要件や最新情報を抽出できていない回が見られた。例えば、「通知機能」は、除外→再検討→除外(決定)のように仕様変更について話されているが、抽出内容が、「再検討」になっている回が見られた。(図 1)

2.3.3. 構造化の不足

【問題の概要・影響】
生成される文書の構造が試行ごとに異なり、書式に統一性が見られなかった。例えば、タイトルや各章の見出しが異なる、番振りか箇条書きかが異なる、「○○する」や言い切りなど文末が異なる、などが確認された。

これらの問題によって、議論内容とは異なる仕様や不必要な機能な実装につながる、開発チームとクライアントで認識の差異が生まれる、開発の手戻りやスケジュールの遅延等につながる、などの問題が起こりかねない。現状の生成 AI をそのまま使うのでは、限界があると判断し、いくつか補正を行う必用があると考える。

3. 提案手法

本研究では、2 章で明らかになった生成 AI の問題に対し、段階的な処理によるアプローチが有効であると考え、以下の手法を提案する。

3.1. 問題解決のアプローチ

【基本方針】
GPT-4o の自然言語理解能力を活用しつつ、その不安定性と不正確性を技術的に補完することで、実用的な議事録自動処理システムを構築する。

【問題別解決戦略】
《出力のばらつき》
素の GPT-4o では、同一入力に対し 5~8 項目と大きな変動があり、平均 6.2 項目と期待値を下回る結果となった。
これに対し、以下の 2 つの技術的アプローチで安定化を図る:

1. 明確な分類基準を提供
   各発言に明確なラベル属性を付与することで、GPT-4o の判断基準を統一する。
   また、決定事項と非決定事項、ノイズを識別し、不要な情報を除去する。
2. 温度パラメータの最適化
   温度パラメータとは、GPT などの生成 AI において、出力のランダム性を制御するパラメータのことである。

こちらを指定することで、一貫性のある分析結果を実現する。

《文脈情報の欠落》
通知機能のような「除外→再検討→除外(決定)」のような複雑な議論変遷に対し、時系列順序と分類ラベルを活用したシンプルな文脈保持方式を採用する。
ここでは、ラベル属性を付与した分類結果を JSON 形式で中間保持する方法をとる。これによる目的は、各発言の元内容と分類結果の対応関係を保持すること、”決定事項”ラベルの発言を確実に特定すること、処理過程の追跡とデバッグを可能にすることである。

《構造化の不足》
試行ごとに書式構造が異なっていた問題について、テンプレートを固定し、完全に統一する仕組みを用いる。本研究では、Jinja2 テンプレートエンジンを用いている。これにより、章立て・項目名の標準化、記述形式の統一が期待される。

【2 段階処理フロー】
2 章の実験にて、生成 AI は、一度に命令を与えると安定性が低下することがわかった。そこで、生成 AI を使用する手順ごとにステップを分けて行う。本研究では、以下の 2 ステップに分けて設計している。

1. 分類段階
   議事録入力 ⇒ GPT-4o 分類 ⇒ 分類結果保存
2. 生成段階
   分類結果読込 ⇒ 決定事項抽出 ⇒ テンプレート適用 ⇒ 仕様書出力

明確に責任分離することにより、各段階での問題切り分けと品質確保を実現する。
【システム構成図】

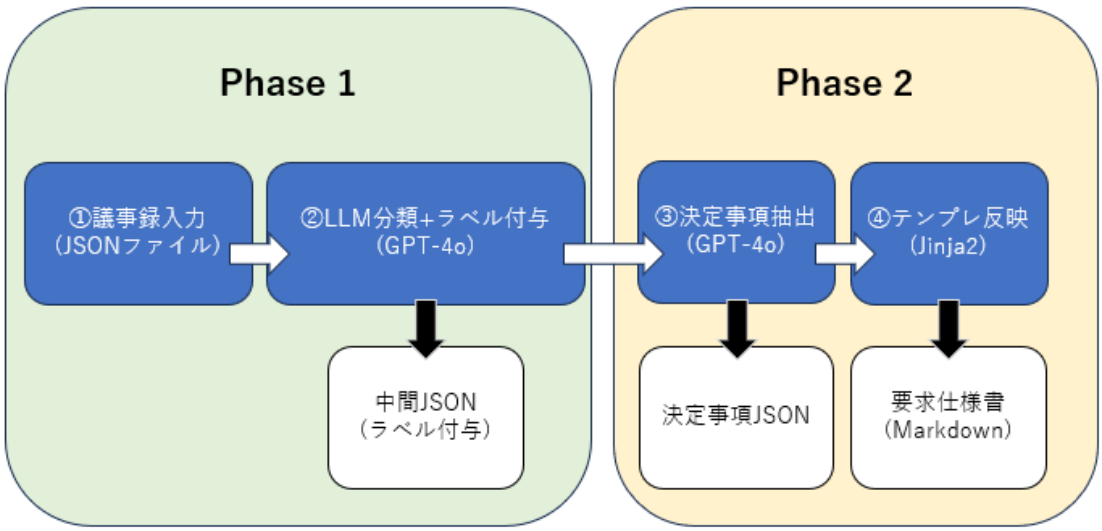

図 4 本研究 システム構成図

3.2. システム設計

提案システムは、図 4 で示した 2 段階処理アーキテクチャを採用する。
下記、3.2.1 は、Phase1、3.2.2 は、Phase1+Phase2-③、3.2.3 は、Phase2-④にそれぞれ該

当する。

3.2.1. 発言分類の安定化

【分類システムの設計】
GPT-4o に明確な分類基準を提供し、判定の一貫性を向上させる。
分類ラベルを”決定事項”を含む 5 分類として、最小限かつ広範囲で分類をかけることで、確認や質問を明確にして、決定事項と検討事項を識別する、雑談とその他を設けることで、ノイズが決定事項に影響を与えることを防ぐ目的がある。
ここで、タイムスタンプを残すことで、時系列関係を追跡できるようにする。
また、ここで JSON での出力を指示し、形式を設計プロセスに合うように調整する。

≪命令プロンプト≫

```
"あなたは議事録を分類する AI です。¥n"
"各発言を次の 5 つのラベルで分類し、指定スキーマの JSON のみを返してください。¥n"
"- decision: 方針/仕様が明確に決まった発言¥n"
"- proposal: 提案・要望¥n"
"- question: 確認・質問¥n"
"- chitchat: 雑談¥n"
 "- other: 上記以外¥n"
"topic は短い名詞句（例: 認証/通知/画面/権限/運用/表示/文言/性能 等)。該当が無ければ「その他」。¥n"
"reason は短い根拠。¥n"
"回答は、必ず日本語で返すようにしてください。"
```

【データ保存形式】
分類結果は以下の構造で JSON ファイルに保存される:

```
[
 {
  “speaker”: “PM”,
  “text”: “では、ログイン初期表示は一旦の方針ということで。”,
```

```
  “timestamp”: “2024-11-20T13:02:00”,
  “label”: “decision”
 }
]
```

3.2.2. 文脈保持メカニズム

【シンプルな文脈保持設計】
複雑なグラフアルゴリズムではなく、以下の実用的なアプローチを採用:

1. 分類ラベルによる情報整理
   ②によりラベルを付与した中間 JSON より、決定事項ラベルのみを確実に抽出し、雑談やその他ラベルなどのノイズの除去を行う。
   また、各分類の発言元の情報を保持することで追跡を可能とする。
2. 時系列情報の追跡
   各発言にて、タイムスタンプを追跡することで、最新情報を特定する。
   これにより、変更前の情報を抽出することを防ぐ。
   また、出力ファイルを確認することで、デバッグを可能としている。
3. 発言者情報の保持
   各発言の発言者を識別することで、抽出項目の発言元・発言者を追跡可能とし、意思決定の主体を明確化します。

【処理の透明性確保】
生成 AI を用いた出力をすべて JSON 形式で保持することにより、情報の追跡やデバッグ・トラブルシューティングを容易にさせる。
また、研究における人手作成との比較を効率的に行う目的もある。

3.2.3. 構造化テンプレート

【テンプレート設計】
Jinja2 を使用し、一貫した文書構造を以下のように定義する

```
## 1. 機能要件
```

```
{% for r in fr %}

- **{{ r.id }}** [{{ r.feature }}] {{ r.statement }}

    - 受け入れ条件:

    {% if r.acceptance_criteria and r.acceptance_criteria|length>0 %}

    {% for ac in r.acceptance_criteria %}

        - {{ ac }}

    {% endfor %}

    {% else %}

        - （未定義）

    {% endif %}

    - 優先度: {{ r.priority }} / 状態: {{ r.status }}

    - 根拠: {{ r.source.speaker }} {{ r.source.timestamp }}

{% endfor %}

{% else %}

- （該当なし）
```

【出力形式の統一】
Jinja2 を用いることで、章立てや項目記号といった書式の統一化を実現する。ここでは、Phase2-③で出力したデータをもとに、決定事項をテンプレートに反映させる。ファイル形式は、仕様書に合わせ、Markdown 形式を指定する。

3.3. 実装詳細

【開発環境・依存関係】

表 2 開発環境・依存関係と使用バージョン

Python	3.14.0
openai	2.16.0
python-dotenv	1.2.1
Jinja2	3.1.6

【技術的特徴】

- API キーの環境変数管理(.env ファイル)
- 堅牢な JSON 解析とエラーハンドリング
- ファイルベースのデータ永続化
- UTF-8 エンコーディング対応

4. 実験と評価

4.1. 実験設定

3 章で提案したシステムの有効性を検証するため、以下の実験を実施した。
ただし、実験データセットが 2 つと少ないため、同一データセットでの性能検証に加えて、データセットをまたいだ汎化検証を含めることで評価の妥当性を高める。

【実験目的】
提案システムが 1.2 で挙げた問題を解決できているかを該当する対象ごとに評価する。

【実験データセット】
DS1: Web アプリケーション開発

規模	35 発言
参加者	4 名
期間	単一セッション（同日 45 分間）
特徴	短時間での要件変更、明確な決定プロセス
決定事項	9 項目
議論変遷	通知機能 「除外→再検討→除外(決定)」

DS2: ホテル・旅館業向けシステム開発

規模	48 発言
参加者	4 名
期間	複数日に跨る議論（約 1 週間）
特徴	長期間での集中的議論、複雑な機能要求
決定事項	14 項目
議論変遷	英語表示「追加→見送り」 緊急案内の仕様変更

DS1 では、「短期間で決定が集中」、DS2 では、「複数日で変更が混在」という違いを持つため、同一条件で両方に一定の性能が出るかを評価することで、提案システムの汎化性を検証できる。

4.2. 実験方法

本研究では、LLM の出力のばらつきを考慮しつつ、temperature の調整→調整結果の凍結(lock 化)→固定条件での再現性評価という手順を採用する。凍結ファイルを用いることで、評価時に「対象物に合わせた都合の良い調整」を排除した汎化検証が可能となる。

【用語定義】

調整	調整対象データセット上で temperature を探索し、最終採用値を決定する工程
凍結	決定した temperature と補正設定・テンプレートを固定し、評価時に変更できない状態にする工程
評価	凍結した条件のまま実行し、決定を統計的に変更する工程

【凍結ファイルの内容】

凍結ファイルは、評価の実現性を担保するために以下を固定する。

- temperature 値
- 補正設定ファイル: 分類する語彙の補正、除外語、ルール類を持つファイル

以降、DS1 から生成した凍結ファイルを”lock_ds1”、DS2 から作成した凍結ファイルを”lock_ds2”と呼称する。

【temperature の調整手順】

temperature は、一般に値が大きい程出力のランダム性が大きくなる傾向にある。本研究では、議事録からの要件抽出において、仕様書としての統合性や再現性を重視するため、

過度なばらつきを避け、一定の柔軟性を確保できる範囲として、0.0~1.0 を探索範囲とし、この範囲を 0.1 刻みで網羅的に検証した。
なお、temperature の決定は、調整対象データセットのみを用いて行う。評価対象データセットの結果を見て温度を選び直すことは行わない。

(1) 候補温度ごとの凍結ファイル作成
temperature を小数第 1 位刻み(0.0, 0.1,…, 1.0)で変化させて調整対象データセットの凍結ファイルを 1 回ずつ作成する。

(2) 候補 temperature ごとの評価
(1)で作成した各凍結ファイルを用いて、LLM 出力のランダム性を考慮して、同一データセット検証を 10 回試行し、表 3 で示す評価指標に基づき記録する。ここでいう評価指標は、temperature の候補間で相対比較できる指標であればよく、以降の最終評価で用いる指標と同一である必要はない。

表 3 評価指標

指標項目	概要
最小抽出項数	10 回の試行中の最小抽出数
最大抽出項数	10 回の試行中の最大抽出数
平均抽出項数	10 回の試行の平均抽出数
平均抽出率	平均抽出項数/全項目数の割合
1 回あたりの総所要時間(最小)	凍結ファイルを用いて評価する際の最小所要時間
1 回あたりの総所要時間(最大)	凍結ファイルを用いて評価する際の最大所要時間
1 回あたりの総所要時間(平均)	凍結ファイルを用いて評価する際の平均所要時間

(3) ピーク温度の決定と最終 lock の確定
(2)で得られた指標のうち、平均抽出率を総合的に見てピークとなる温度を採用する。
ここでピークとは、要件抽出の安定性と抽出性能のバランが最も良い点を指す。
ここで採用した温度に対して、汎化検証で用いる凍結ファイルとして保存する。

(4) 2 つのデータセットで同様に実施
(1)~(3)の手順を、DS1, DS2 それぞれで行い、採用凍結ファイル”lock_ds1”, “lock_ds2”を作成する。

【評価の実行条件】
作成した凍結ファイルを用いて、以下の 4 条件で評価する。

調整対象	評価対象	凍結ファイル	目的
DS1	DS1	lock_ds1	性能検証
DS1	DS2	lock_ds1	汎化検証
DS2	DS2	lock_ds2	性能検証
DS2	DS1	lock_ds2	汎化検証

これにより、「対象物に合わせた補正」ではなく、それぞれの調整がもう片方に適用した結果として汎化性を示す。

【所要時間】
実用性評価の一部として、所要時間を以下の粒度で測定する。

- データ読み込み開始→ラベル付与した中間 JSON 出力まで
- 中間 JSON 読み込み開始→決定事項 JSON 出力まで
- 決定事項 JSON 読み込み開始→仕様書出力まで
- 上記を参照し、総所要時間を計測する

4.3. 評価方法

本研究では、以下の 2 つの比較評価により、提案システムの有効性を検証する。
【評価対象 1: 素の生成 AI vs. 本研究プログラム】
2 章で観測された問題点に対し、本研究プログラムがどの程度改善できているかを評価する。

最小抽出項数	10 回のうち、最も少なかった抽出数
最大抽出項数	10 回のうち、最も多かった抽出数
平均抽出項数	10 回の抽出平均数
平均抽出率	平均抽出項数/全項目数の割合
書式の統一性	Jinja2 テンプレートエンジンの有用性検証
所要平均時間	２つを比較し、どれほどの差があるか

【評価対象 2: 本研究プログラム vs 人手作成】
実運用上の実用性を評価し、人手作成に対する優位性と課題を明確化する。

最小抽出項数	10 回のうち、最も少なかった抽出数
最大抽出項数	10 回のうち、最も多かった抽出数
平均抽出項数	10 回の抽出平均数
平均抽出率	平均抽出項数/全項目数の割合
重要項目の抽出率	重要な項目が漏れていないか

所要平均時間	２つを比較し、どれほどの差があるか

【測定手順】

(1) 素の AI と本研究プログラムは、各条件で 10 回実行し平均を比較する

(2) 人手作成はベンチマークとして、1 回実施し、参照値として比較する

(3) 上記の評価項目に基づき、技術的改善効果と実用性の両面から提案システムの価値を示す

4.4. 実験結果

4.4.1 temperature 調整結果

表 4 DS1 temperature 測定結果

temperature	最小抽出数	最大抽出数	平均抽出数	最小所要時間	最大所要時間	平均所要時間
0.0	6	10	7.1	1m34s	1m56s	1m44s
0.1	6	9	7.4	1m29s	1m56s	1m43s
0.2	8	9	8.2	1m30s	1m59s	1m43s
0.3	7	10	8.1	1m32s	2m02s	1m47s
0.4	7	10	7.9	1m25s	2m03s	1m44s
0.5	6	11	8.2	1m35s	1m49s	1m43s
0.6	5	11	7.5	1m29s	1m46s	1m34s
0.7	5	11	8.2	1m30s	1m52s	1m46s
0.8	5	11	8.5	1m33s	1m55s	1m44s
0.9	4	12	8.4	1m36s	1m53s	1m48s
1.0	5	12	8.7	1m32s	1m56s	1m43s

表 5 DS2 temperature 測定結果

temperature	最小抽出数	最大抽出数	平均抽出数	最小所要時間	最大所要時間	平均所要時間
0.0	8	10	8.5	1m34s	2m06s	1m55s
0.1	8	10	8.3	1m29s	2m10s	1m54s
0.2	7	11	8.8	1m30s	2m02s	1m54s
0.3	8	13	9.2	1m32s	2m02s	1m40s
0.4	7	12	9.0	1m25s	2m03s	1m51s

0.5	9	12	10.1	1m35s	1m59s	1m42s
0.6	8	14	10.9	1m29s	2m01s	1m57s
0.7	8	14	12.2	1m30s	1m52s	1m44s
0.8	6	15	13.6	1m33s	2m15s	1m53s
0.9	8	14	12.4	1m36s	2m13s	1m49s
1.0	8	15	13.6	1m32s	2m06s	1m46s

表 4, 5 より、DS1 では、temperature=0.2, DS2 では、temperature=0.6 を採用することとする。これは、抽出率だけではなく、不要な情報や AI の予測推論の影響が最も小さい値を選択した。以下、表 6,7 は、採用した凍結ファイルを用いた結果である。

表 6 Lock_ds1_DS2 比較結果

DS	最小抽出数	最大抽出数	平均抽出数	最小所要時間	最大所要時間	平均所要時間
DS1	8	9	8.2	1m30s	1m59s	1m43s
DS2	8	11	9.2	1m37s	2m12s	1m59s

表 7 Lock_ds2_DS1 比較結果

DS	最小抽出数	最大抽出数	平均抽出数	最小所要時間	最大所要時間	平均所要時間
DS1	7	13	9.6	1m26s	2m07s	1m47s
DS2	8	14	10.9	1m29s	2m01s	1m57s

4.4.2 素の生成 AI vs 本研究プログラム

【データセット 1: 比較結果】

表 8 DS1 評価対象 1 結果

対象	最小抽出数	最大抽出数	平均抽出数	平均抽出率	書式の統一性	平均所要時間
素の生成 AI	5	8	6.2	68.89%	×	2m16s
本研究	8	10	8.4	93.33%	○	2m04s

【データセット 2: 比較結果】

表 9 DS2 評価対象 1 結果

対象	最小抽出数	最大抽出数	平均抽出数	平均抽出率	書式の統一性	平均所要時間
素の生成 AI	7	12	9.7	69.29%	×	2m49s
本研究	9	14	10.7	76.43%	○	2m43s

4.4.3 本研究プログラム vs 人手作成

本節では、提案システムが「人手作成にたいして実用上どの程度代替になり得るか」を評価する。比較対象は、人手で作成した決定事項リスト(DS1: 9 項目、DS2: 14 項目)とし、提案システムの出力結果がこれらをどの程度含むかと作成に要する所要時間を中心に比較した。

【データセット 1: 比較結果】

表 10 DS1 評価対象 2 結果

対象	最小抽出数	最大抽出数	平均抽出数	平均抽出率	重要項目の抽出率	平均所要時間
本研究	5	8	6.2	68.89%	87%	2m16s
人	9	9	9	100%	100%	196m27s

【データセット 2: 比較結果】

表 11 DS2 評価対象 2 結果

対象	最小抽出数	最大抽出数	平均抽出数	平均抽出率	重要項目の抽出率	平均所要時間
本研究	8	15	10.7	76.43%	84%	2m16s
人	14	14	14	100%	100%	213m42s

4.4.4 総合評価

素の AI 出力と比較して、本研究プログラムの抽出率の方が高いことから、温度パラメータの最適化、命令分けのフェーズ設計は効果的であることが分かった。

温度パラメータに限らず、AI の予測推論が加わることが確認されたが、温度パラメータが大きい程、その影響が大きく見られた。
しかし、本研究プログラムのすべての出力結果にて、日本語と英語が混ざった出力が見られたことから、テンプレートの指定のみでは不足であることが確認された。
また、各フェーズでの所要時間については、本研究プログラムの全試行において、ラベル付けは 15 秒前後、仕様書出力は、1 秒未満と共通であり、所要時間のほとんどが決定事項の抽出処理であることがわかった。
さらに、抽出率が 100%に到達しないケースに加え、人手の決定事項数を上回る抽出（過抽出）も確認された。これは、同義表現の重複、1 項目の粒度分割、ログに存在しない推測補完が混在したためと考えられる。
また、抽出の欠落の傾向から現在の生成 AI は、特定の要素に対して分析しているのではないと考える。

5. 考察

5.1. 言語の混雑

OpenAI は、アメリカの企業であり、GPT-4o を含む生成 AI は、英語で学習されていると考えられる。その結果、抽出ファイルの内容が英語で記されるケースが多く見られた。これは、命令に「日本語で返すこと」を条件に加えて行っても改善されなかったことから、別のアプローチを必要とすることがわかった。

5.2. 実行時間の違い

今回使用している API は、外部のものになり、世界中の多くのユーザーが使用するものである。そのため、使用 PC のスペックや API のアクセス状況によって、多少の差が生まれるのだと考えるが、生成 AI を用いることで、人が作成するよりはるかに早いことは間違いない。

5.3. 生成 AI の現状と問題

本研究では、素の生成 AI 出力に対し、分類・正規化・テンプレート反映により品質は向上したが、補正を加えても抽出率 100%には達しないこと、全項目数を上回る過抽出が発生することが確認された。これは、議事録ログが暗黙的合意や文脈依存を含み、決定事項が明示されない場合がある点に加え、生成 AI が本質的に確率的生成であり、与えられた情報を忠実に抽出するというより、整合的な文章を生成する方向に最適化されていること

が原因であると考えられる。
将来モデル(GPT-5, GPT-6o など)により指示追従性や誤った解釈の頻度は改善し得るといえる。しかし、確率的生成である限り誤りをゼロにすることは難しい。加えて、生成 AI の進化に伴い、膨大なデータを用いた学習により、予測推論の能力向上が考えられ、これによる影響で、議事録にある情報だけでなく拡大解釈し、過抽出につながりかねない。
したがって、現状の生成 AI を用いて実運用で人の代替を目指すには、根拠付与の強制、スキーマ検証、重複除去等の制約や検証機構だけではなく、必要に応じた人手の確認や生成 AI の特性に人が合わせる必要がある。

5.4. 本プログラムの手ごたえ

提案システムの本質的な価値は、「生成 AI の出力をそのまま採用する」のではなく、要件抽出に必要な処理単位へ分解・整理したうえで生成 AI を扱う点にある。
実験結果から、所要時間の短縮は顕著に表れ、会議直後に仕様書ドラフトを提示できる速度感としては申し分ないと判断する。しかし、要件の抽出の確実性や AI の予測推論の除去といった課題を完全に満たすものではないため、今後の課題として残る。
AI がテキストを読み学習されたデータを用いて分析するのに対し、人が雰囲気や感覚といったあいまいな情報でもコミュニケーションが取れていることから、「決定」や「不要」といった明確な単語を含まない議事録に対しては精度が落ちることが予想される。

5.5. 実用化への示唆

実用化を想定した場合、提案システムの以下の点で有効である。

- 仕様書ドラフトを短時間で生成し、PM/開発者がレビューして確定する運用が可能
- 中間出力ファイルが証拠として残るため、根拠の追跡が容易
- テンプレート差し替えにより、要件仕様書だけでなく、様々な文書に展開しやすい

ただし、使用する AI モデルやプロンプトの変更により、出力特性が変化する可能性を考慮しなければならない。

6. 課題・まとめ

6.1. 課題

本研究の制約及び今後の課題は、以下の通りである。

① データセット規模の制約
2 プロジェクトでの評価であり、議事録の種類(組織文化、発言の粒度、参加者

数)を十分に網羅できていない。
生成 AI の特性に合わせたデータセットの記述修正を行う。

② 最終状態判定の難しさ
議論変遷のあるケースで、途中経過情報ではなく、最終議論情報を残すことが重要だが、発言表現があいまいな場合、誤判定の余地が残る。

③ 評価の限界
抽出率は、人手決定事項に依存するため、人手側の定義揺れがあると評価がぶれえる。今後は複数人アノテーション等により、ベースラインの信頼性を上げる必要がある。

④ コスト・セキュリティ
外部 API 利用を前提としているため、コスト最適化と機密情報の扱い(匿名化・マスキングなど)が実運用上の要点となる。

6.2. まとめ

本研究では、議事録ログから最終要求要件を抽出し、構造化テンプレートにより仕様書として出力するする手無を提案した。生成 AI 単体利用で観測された課題に対し、分類・正規化・テンプレート生成の段階処理を導入することで出力の安定化と確認コストの低減が実現できた。

実験では、異なる性能を持つ 2 つのデータセットに対し、人手作成と比較し、所要時間の大幅な短縮を確認したが、一定水準の抽出率や AI の特性の緩和を実現することができなかった。今後は、最終状態判定の精度向上、評価基準の厳密化を進め、より汎用的な議事録ベース仕様書生成へ発展させる。

7. 参考文献

[1] OpenAI API 公式ガイド

- API Reference https://platform.openai.com/docs/api-reference/introduction
- Platform Quickstart https://platform.openai.com/docs/quickstart
- OpenAI API 利用料金: https://openai.com/ja-JP/api/pricing/
-

[2] Jinja 公式 Docs https://jinja.palletsprojects.com/en/stable/

[3] Taohong Zhu, Lucas C. Cordeiro, Youcheng Sun, “ReqInOne: A Large Language Model-Based Agent for Software Requirements Specification Generation,” arXiv, 2025. https://arxiv.org/pdf/2508.09648

[4] D. Milchevski ほか, “Multi-Step Generation of Test Specifications using Large Language Models …,” ACL Industry, 2025. https://aclanthology.org/2025.acl-industry.11.pdf