

From Prompts to Templates: A Systematic Prompt Template Analysis for Real-world LLMapps

Yuetian Mao, Junjie He, Chunyang Chen

FSE Companion '25: Proceedings of the 33rd ACM
International Conference on the Foundations of Software
Engineering

S2310157 谷口

背景：LLMapps

LLM主導型のアプリ（LLMapps）の普及

- LLMapps : LLMが組み込まれたアプリ
- GPT-4, LLaMAなどの高性能なモデルの登場により広まる
- LLMappsの特徴
 - 主にLLMがユーザからの要求を処理
 - 主目的：LLM利用の円滑化
 - ⇒ アプリ自体は軽い実装（ローコード）
 - ユーザがLLMを容易に扱えるようにする
（プロンプトテンプレート）

背景：プロンプトテンプレート

曖昧な指示をテンプレートから再構成

- ・ ユーザの曖昧な指示を明確な指示に再構成するテンプレート
- ・ 静的なテキストと動的な変数（プレースホルダー）で構成
 - 静的なテキスト：LLMの役割や出力形式の指定など不変の指示
 - プレースホルダー：ユーザ入力や外部知識など変化する指示

プレース
ホルダー

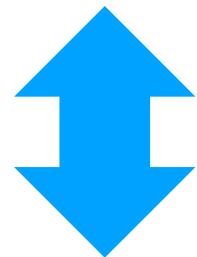
あなたは音楽推薦システムです。あなたは
{ユーザー要件} に最適な曲の推薦を返答します。
JSON オブジェクトのみを返してください。他のものは送る必要はありません。
作り話をしないでください。
JSON の応答には title、artist、album、year を含めてください。次の
JSON 形式を使用してください：
{
 "songs": [
 {
 "title": "曲名 1",
 "artist": "曲 1 のアーティスト",
 "album": "曲 1 のアルバム",
 "year": "リリース年"
 }
]
}

本論文 Fig 1から抜粋、改変

背景：本論文の目的

プロンプトテンプレートの課題

- 有名なモデルプロバイダの大半がこの機能をサポート
- LLMappsの根幹である重要な機能



- 高品質なテンプレートの作り方については手探り
- 体系的な分析、理論が存在しない

⇒実際に使用されているプロジェクトから

プロンプトテンプレートを収集し体系的な分析を行う

本論文のRQs

プロンプトテンプレート分析

- RQ1 : プロンプトテンプレートを構成する一般的なコンポーネントは何か？ その中に頻出するパターンは？
- RQ2 : プロンプトテンプレート内のプレースホルダーの種類や位置にはどのようなものがあるか？

プロンプトテンプレート検証

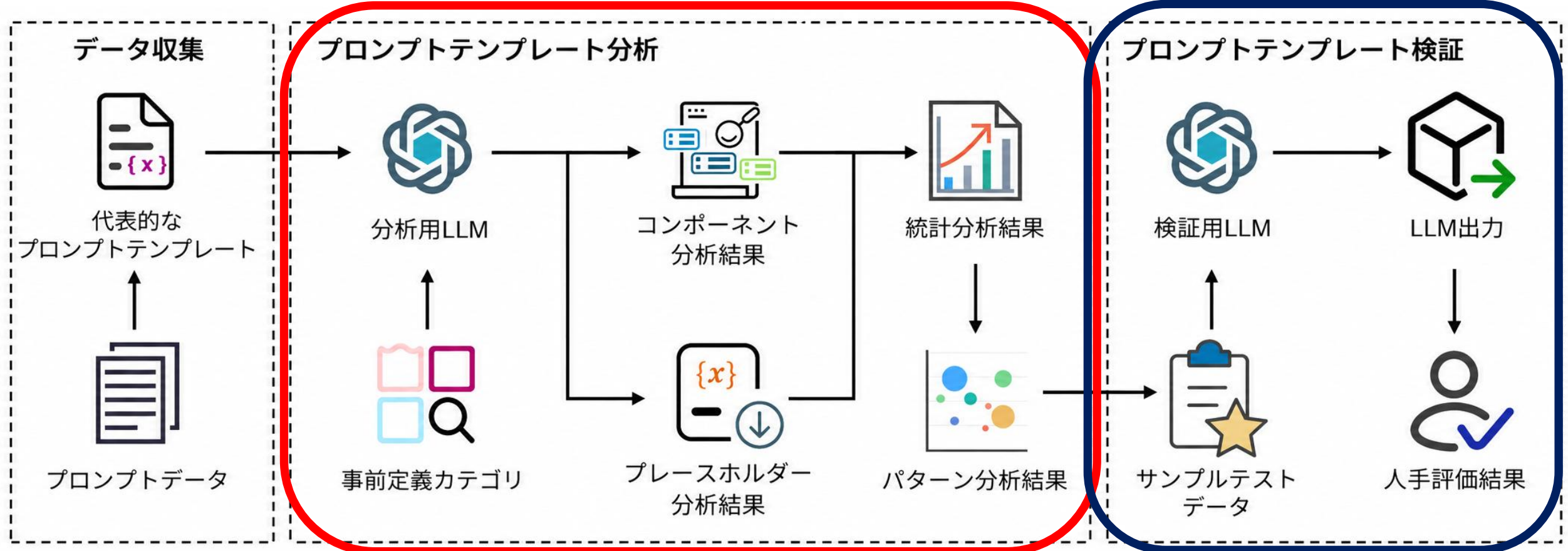
- RQ3 : プロンプトテンプレートの構成は、LLMの指示追従能力にどのような影響を与えるか

研究全体の流れ

全体のパイプライン

RQ1, 2

RQ3

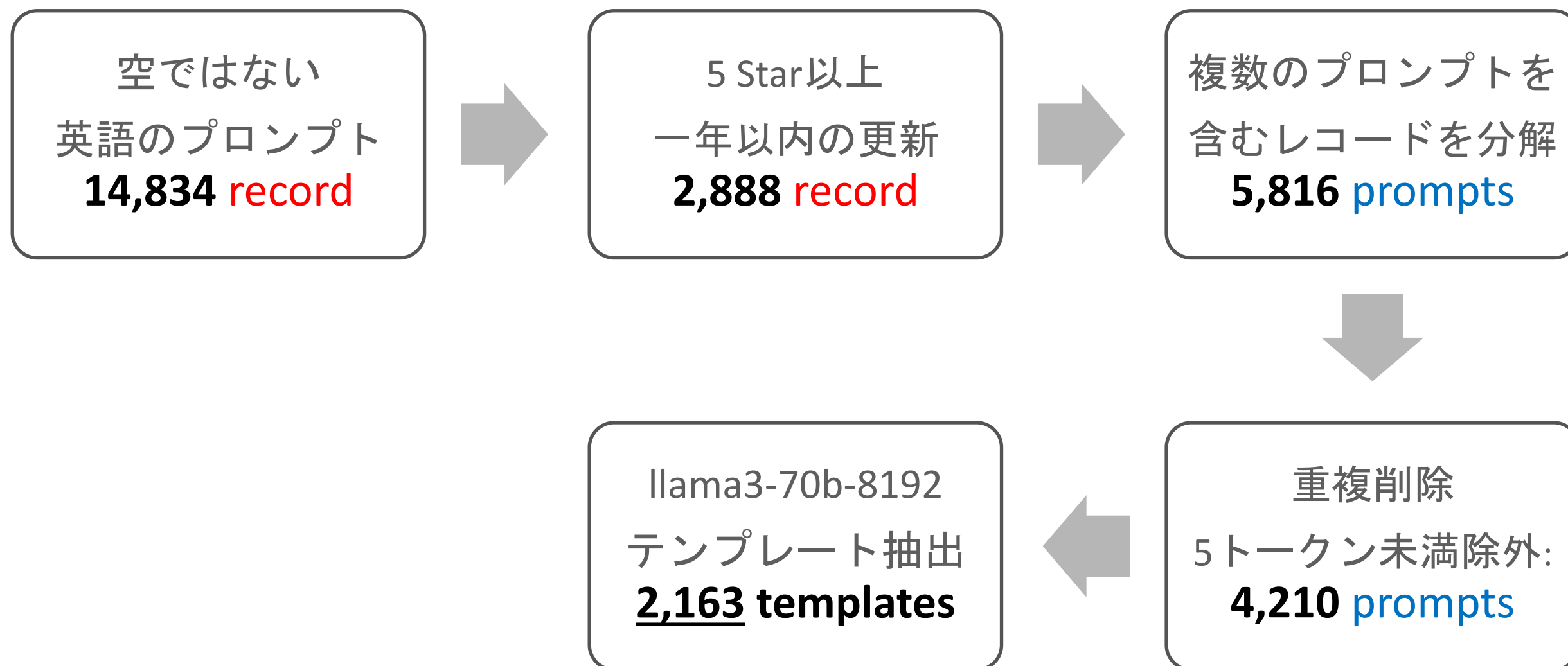


本論文 Fig 2を改変

データ収集

プロンプトテンプレートの選定

- 「PromptSet」データセットを使用
 - 2024/01/10までのGitHubプロジェクトのプロンプトを集めたもの



プロンプトテンプレート分析

コンポーネント分析 (RQ1)

- ・コンポーネントのカテゴリ、配置、内容进行分析

[コンポーネント例]

あなたは音楽推薦システムです。あなたは
{ユーザー要件} に最適な曲の推薦を返答します。
JSON オブジェクトのみを返してください。他のものは送る必要はありません。
作り話をしないでください。
JSON の応答には title、artist、album、year を含めてください。次の
JSON 形式を使用してください:
{
 "songs": [
 {
 "title": "曲名 1",
 "artist": "曲 1 のアーティスト",
 "album": "曲 1 のアルバム",
 "year": "リリース年"
 }
]
}

本論文 Fig 1から抜粋、改変

Profile/Role : 役割 Directive : 指示部

Constraints : 制約 Output Format/Style : 出力形式

プロンプトテンプレート分析

プレースホルダー分析

- 各プレースホルダーの分類、配置、命名を分析

LLMを用いた分析

- 使用モデル
 - コンポーネント分析 : llama3-70b-8192
 - プレースホルダー分析 : gpt-4o
- LLMが全体を分析 ⇒ 全体の一部のみ人間が検証
- 精度はコンポーネント分析(86%)、プレースホルダー分析(81%)
- 全体の分析も8割程度の精度が期待できる

調査結果(RQ1)：コンポーネント

コンポーネントの分類、位置

コンポーネント	定義	割合
Profile/Role	モデルの役割、演じるべきキャラクター	28.4%
Directive	プロンプトの主目的、基本的には指示や質問	86.7%
Workflow	モデルが守るべき作業の順序	27.5%
Context	モデルが参照すべき背景情報や文脈	56.2%
Examples	どのように出力すべきかの例	19.9%
Output Format/Style	出力形式の指定	39.7%
Constraints	出力時にモデルが守るべき制約	35.7%

本論文 Table 3を改変

- ・ コンポーネントは主に7種類
- ・ Profile/Roleは先頭、Examplesは最後尾に置かれることが多い

調査結果(RQ1)：コンポーネント

コンポーネントの詳細

- Directiveは指示文の方が質問文より多く使われていた
- Output Style/Formatで最も指定されていたのはJSON形式
- JSON形式の指定には次の3パターンが確認された
 - パターン1：出力がJSON形式であることのみを示す
 - パターン2：パターン1 + 含める属性名（キー）をリストアップ
 - パターン3：パターン2 + 各属性名の詳細な説明の追加
- Constraintsは以下の3種類が一般的
 - 「除外」（46.0%）
 - 「包含」（35.6%）
 - 「文字/単語数」（10.5%）

調査結果(RQ2)：プレースホルダー

プレースホルダーの分類、位置、命名

カテゴリ	定義	例	割合
User Question	ユーザからの質問、要求	{question}, {query}	24.5%
Contextual Information	主題ではないがタスク遂行を補助する背景情報	{chat_history}, {background_info}	19.5%
Knowledge Input	プロンプトが直接処理、操作する主要内容	{document}	50.9%
Metadata/ Short Phrases	特定の条件や目標を示す簡潔な入力、設定	{language}, {username}	43.4%

- プレースホルダーは主に4種類
- Knowledge Inputが最も多く、先頭か最後尾に置かれる
- "text"(4.44%), "input"(2.35%)などの不明瞭な命名が見られる

調査結果(RQ3)：プロンプトの検証

JSON形式の出力の指定方法の比較

	フォーマット追従性		コンテンツ追従性	
	Llama3-70b-8192	gpt-4o	Llama3-70b-8192	gpt-4o
パターン1 Json指定	3.09	3.21	3.70	3.50
パターン2 +属性名の指示	4.66	4.86	4.02	4.30
パターン3 +各属性の 詳細の追加	<u>4.90</u>	<u>4.96</u>	<u>4.47</u>	<u>4.53</u>

- 代表的な5つのテンプレートに対し
3種のJSON形式の指定を適用し、3つの異なる入力で比較
(計45個のプロンプト)
- 人間による1-5の5段階評価でパターン3が最も良い結果

調査結果(RQ3)：プロンプトの検証

Constraintsの除外の効果

- 前述したJSON形式のパターン3の15個のプロンプトで実験
- 以下の除外制約の効果を検証
 - 「JSON形式以外の出カテキストを提供しないこと」
- 出力がJSON形式のみで提供されるかで評価
- llamaモデルでは除外制約によって40% ⇒ 100%の精度に改善
- gpt-4oでは86.67% ⇒ 100%の精度に改善

調査結果(RQ3)：プロンプトの検証

Knowledge Input プレースホルダーの位置

	モデル	コンテンツ追従性 (質問)	コンテンツ追従性 (タスク)
指示先行	LLaMA	4.52	3.72
	GPT	4.61	4.26
プレースホルダー 先行	LLaMA	4.84	4.63
	GPT	4.70	4.60

- ユーザの質問に対し知識を検索してから回答するRAG
- Knowledge Inputは先頭、末尾どちらに配置すべきかを検証
- 指示が後にくるように配置した方が良い結果に
- Knowledge Inputが長くなるほど指示先行の結果は悪化した

示唆

プロバイダへの示唆

- ・ 事前に定義されたテンプレートの提供がAPIのパフォーマンスを向上させる
- ・ テンプレートをテスト改善しやすくする
自動評価ツールを作るべき

開発者への示唆

- ・ プレースホルダーの命名や配置を工夫することで保守性を高くできる
- ・ よく定義されたテンプレートの使用で低コストなLLMを強化
- ・ テンプレート次第ではZero-shotで十分なケースが多い

所感

LLMを分析に使ってるのが面白い

- LLMの分析結果の一部（5%）を検証して全体の分析結果の精度を保証する
- 正直本当に信じられるのかは疑問

弱いモデルもプロンプト次第

- 弱いモデルは複雑な思考は難しいが指示への追従性はプロンプト次第で可能
- Few-shotは強力なプロンプト手法だがタスク次第ではより少ないトークンで済むプロンプト手法が存在する