LLM活用の要点

仕組み・限界・改善策

全体の内容

LLM活用の要点 – 仕組み・限界・改善策

① LLMの仕組み

[入力文] → [次の語を確率予測] → [出力文]

本質的に誤りを含む可能性

② 出力品質を左右する要因

- プロンプトの構成
- 思考モードのON/OFF

使い方次第で品質が大きく変わる

③ RAG — 外部知識で補強

[質問] → 資料検索 → 関連情報抽出 → LLM回答

固有情報・最新情報への対応力を向上

④ 精度とセキュリティの向上策

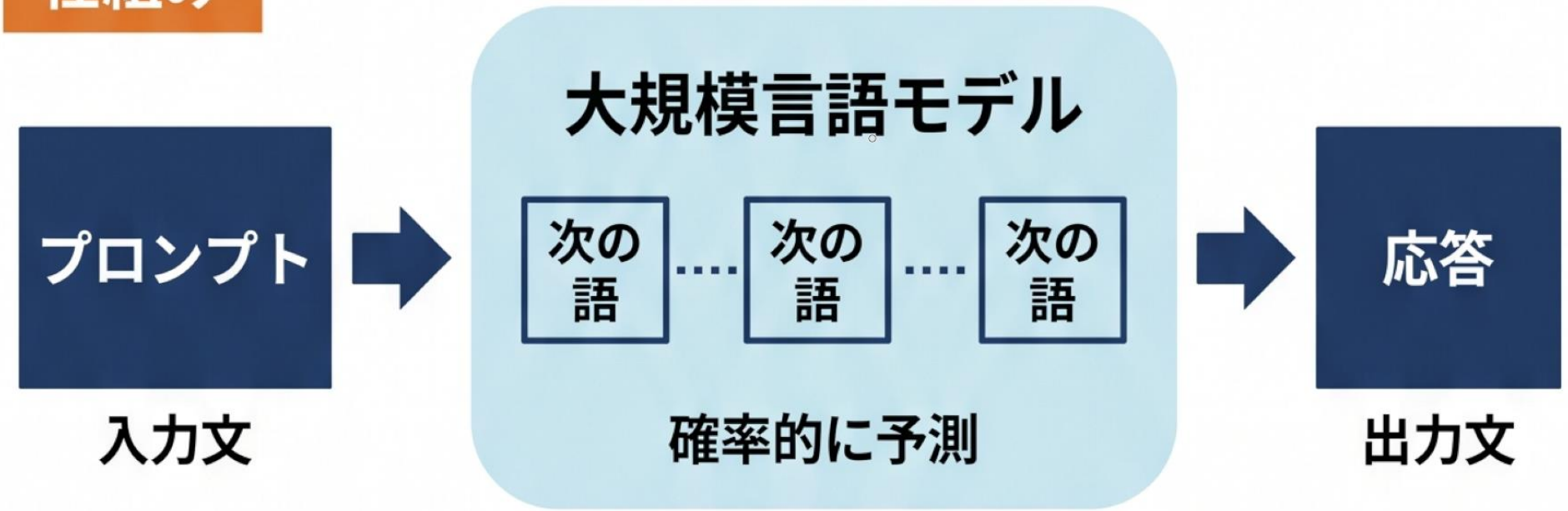
- ✓ ローカル運用
- ✓ 2モデル併用
- ✓ 2段階判定

精度向上 + 情報セキュリティ強化

限界を理解し、適切に補えば強力なツールになる

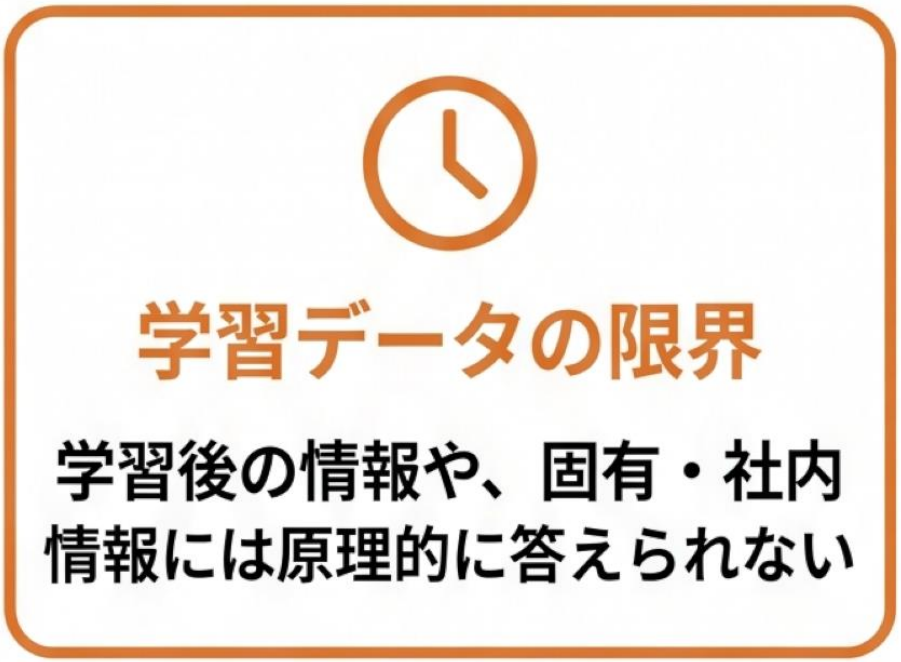
大規模言語モデル（LLM）の仕組み

仕組み



検索ではなく、学習した分布から最も自然な語の連なりを生成

大規模言語モデル（LLM）の本質的限界



本資料の構成

Database Lab.

① プロンプトの設計

② 思考モード
推論の深さを切替

③ 実行環境の選択
クラウド/ローカル

④ RAG
外部知識の補完

⑤ 複数文書の横断処理

⑥ 複数モデルの併用
役割分担

性質を理解した上で、設計・運用で補う

主要概念と用語の整理

本資料で用いる用語

基礎

プロンプト
LLMへの入力文全体。指示・文脈・参考情報・質問から構成される

運用設定

思考モード
回答前に内部で推論ステップを展開。
精度↑／時間・コスト↑

クラウド型／ローカル
クラウド：高性能だが外部送信／
ローカル：機密性高いが規模に制約

拡張手法

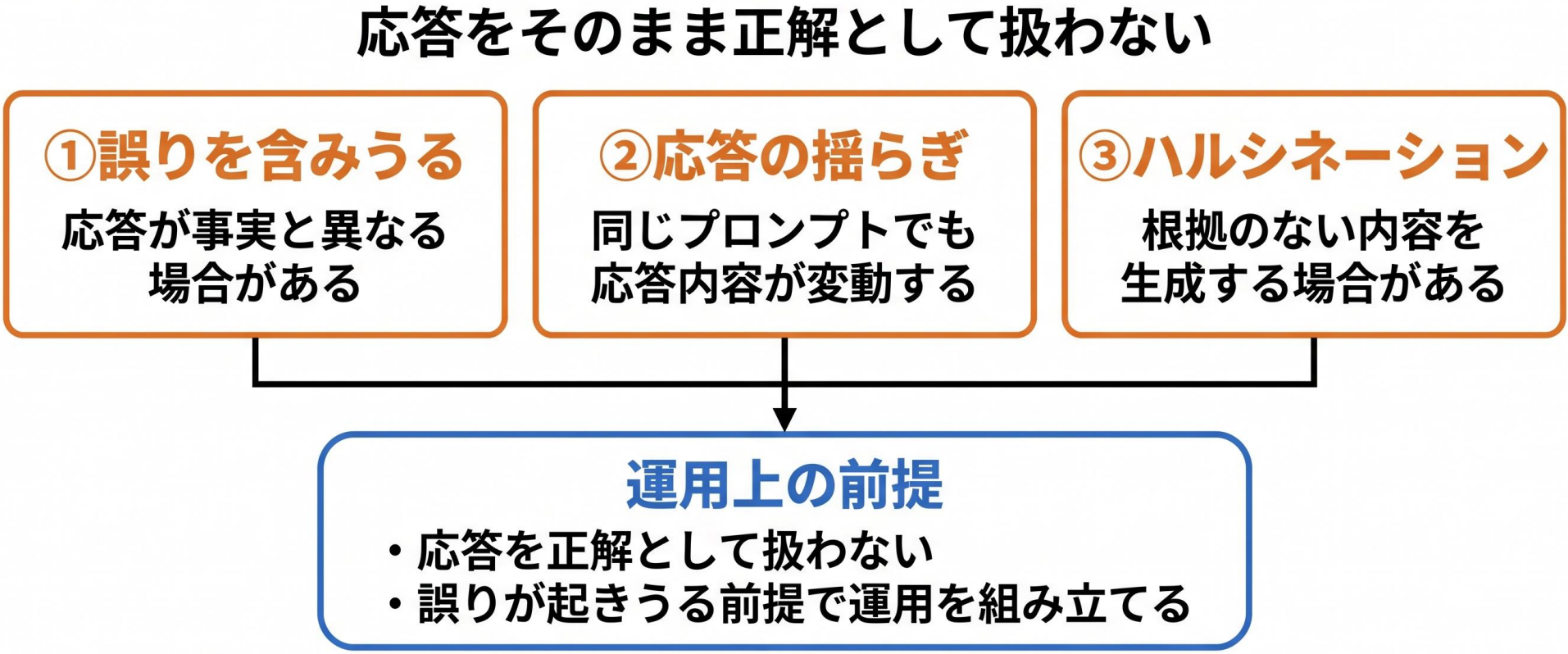
RAG
関連文書を検索しプロンプトに追加。固有・最新情報に対応

冊子横断処理
複数文書にRAGを適用し、文書を跨いで統合回答

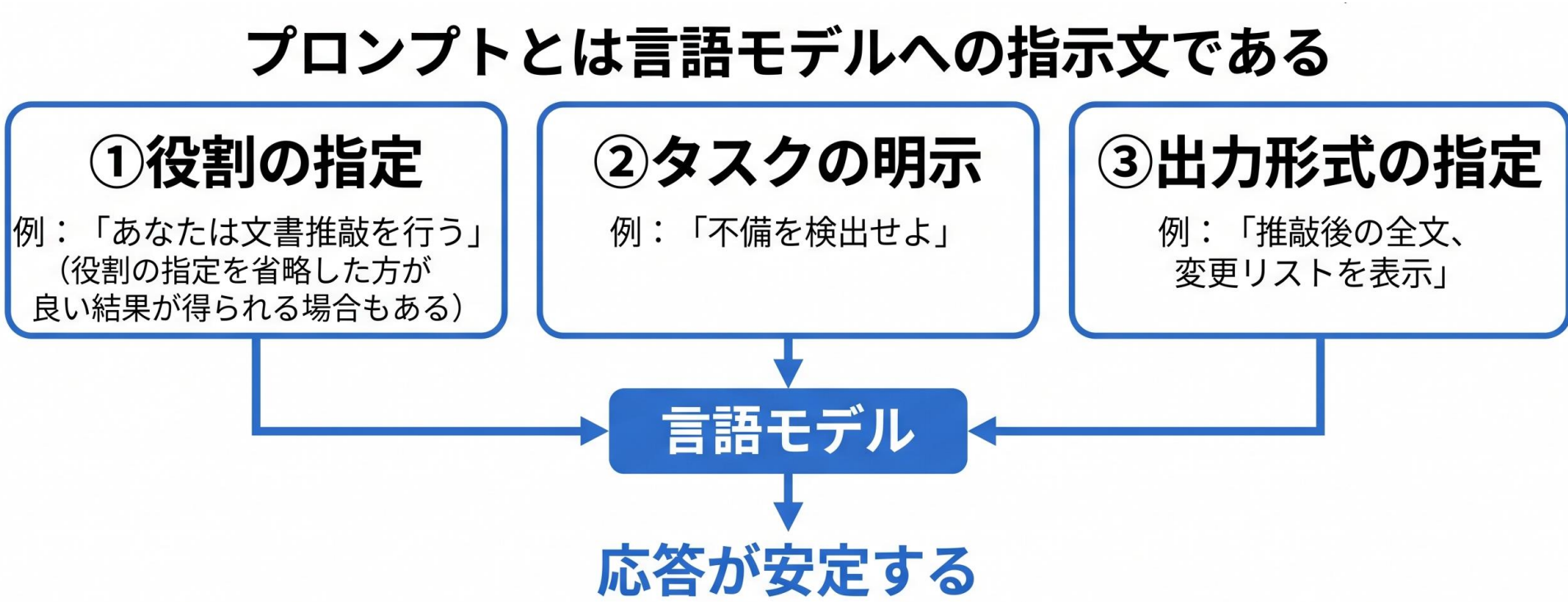
2モデルの併用
用途や負荷に応じて高速・軽量モデルと高精度モデルを使い分け

基礎をふまえ、運用設定と拡張手法を組み合わせる

言語モデルは誤りを含む可能性がある



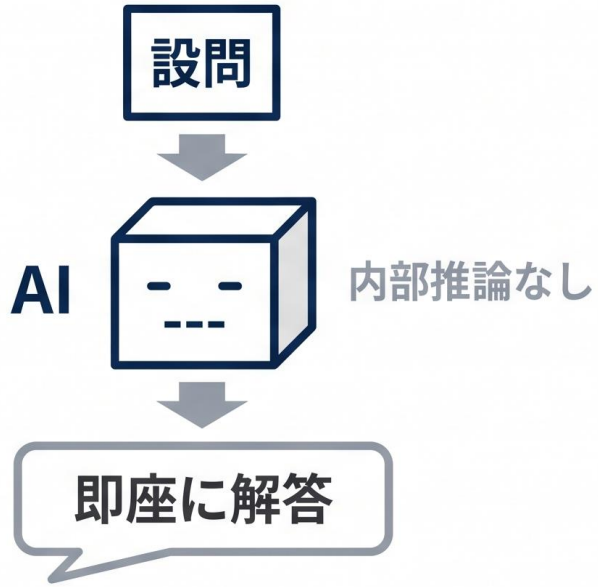
プロンプトの基本構成


プロンプトとは言語モデルへの指示文である
①役割の指定
例：「あなたは文書推敲を行う」
（役割の指定を省略した方が
良い結果が得られる場合もある）
②タスクの明示
例：「不備を検出せよ」
③出力形式の指定
例：「推敲後の全文、
変更リストを表示」
言語モデル
応答が安定する

思考モード

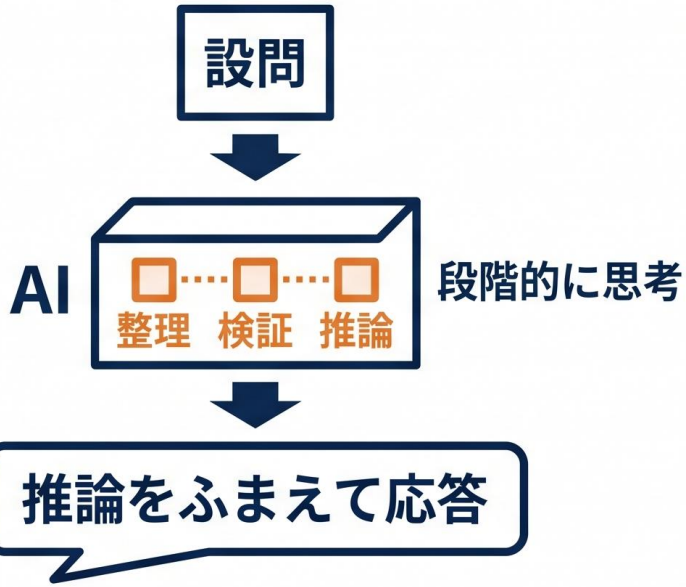
応答前に内部で推論過程を生成する機能

思考モード：オフ



⚠ 設問の不備を見落とす傾向

思考モード：オン



✓ 論理関係の追跡に有効

製品により『推論モード』等の名称でも提供される／論理関係の追跡が必要な場面でオンに

ローカルLLMの仕組み　ー　クラウド型との比較

質問と応答が外部サーバへ送信される

すべて手元のマシン内で処理（外部送信なし）

RAG の使い分け：一般的な質問と、固有・最新情報を必要とする質問

LLM 単体

事前学習データの範囲で応答

適した場面：一般的な質問

RAG(検索拡張生成)

登録文書を根拠に応答

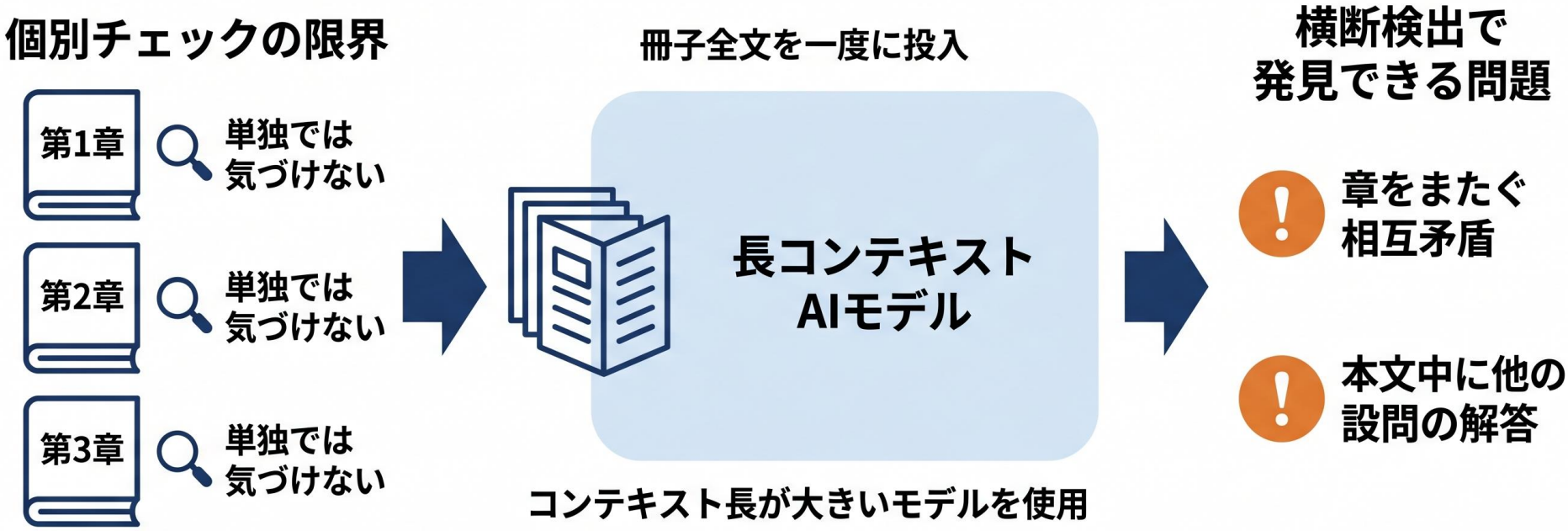
適した場面：自校固有の文書、最新文書

※ ハルシネーション抑制

RAG では、プロンプトの記載に応じて、既存資料の関連個所が検索される。

冊子横断処理

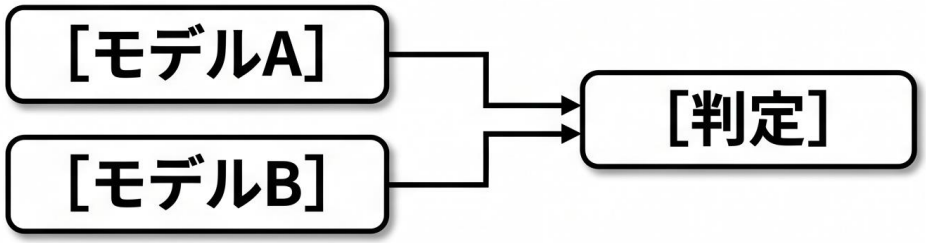
冊子全文を一度にAIへ投入し、相互参照で問題を発見する



複数を並べて初めて検出できる問題がある

２つのモデルの併用

構造や学習データが異なる2つのモデルで照合する



※ 誤り方の傾向が一致しにくいため照合が有効

判定ルール

パターン1	パターン2	パターン3
両者が同一の指摘 →要修正	片方のみの指摘 →要人手/確認	両者とも問題なし

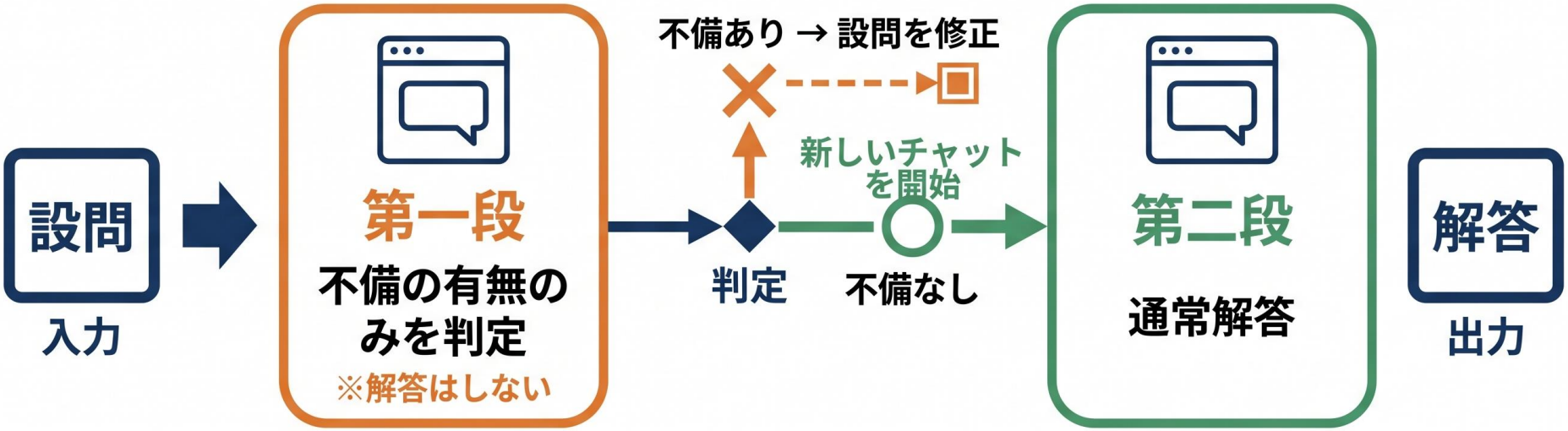
（注意点）

⚠ 両モデルが同じ誤りをする可能性は残る

⚠ クロスチェックは誤りの完全な排除を保証しない

二段階の判定処理

不備判定と解答をチャットを分けて実行



なぜ分離するのか

同一プロンプトで両方を指示すると、
解答指示が優先され不備を見逃しやすい

プロンプト分離の実装

『新しいチャット』を開始すると会話履歴が引き継がれず、第一段の文脈が第二段に影響しない