卒業論文

翻訳エージェントのための構造化プロンプト

指導教官　　村上 陽平 教授

立命館大学 情報理工学部
先端社会デザインコース 4 回生
2600210095-8

加奥 咲江

2024 年度（秋学期）卒業研究 3（CH）
令和 7 年 1 月 31 日

翻訳エージェントのための構造化プロンプト

加奥 咲江

内容梗概

近年，ますますグローバル化が進んでおり，異なる言語や文化間でコミュニケーションを行う機会が増加している．機械翻訳や，文脈を理解できる大規模言語モデル (LLM) の発展に伴い，言語や文化の壁を低くすることが可能になってきた．一方，多義語や文化依存の単語など複数の意味を持つ単語を含む文では，入力に対して一意に出力を返す関数的な機械翻訳は，文脈から語義の曖昧さを解決できなくても翻訳を行うため，適切な翻訳文を生成できない場合がある．この問題を解決するために，ユーザとのインタラクションを通じて翻訳を生成する「翻訳エージェント」が注目されている．例えば，人間の翻訳者が行うように，ユーザに文脈やニュアンスを確認しながら翻訳を行うものが翻訳エージェントである．このような翻訳エージェントを，大規模言語モデル (LLM) を用いて構築するには，ユーザとどのようなインタラクションを行うのかというシナリオを事前にプロンプトとして与える必要がある．しかしながら，自然言語で記述されるプロンプトは曖昧性が高いことに加え，ユーザとのインタラクションが増えるにつれてプロンプトが長くなるため，プロンプトの最初に与えられたシナリオから大規模言語モデルが逸脱する問題がある．

そこで，本研究では，翻訳エージェントを制御するための構造化された YAML のプロンプト記法を提案する．具体的には，シナリオのシンタックス (文法) とセマンティクス (意味) を定義するシナリオの仕様と，その言語に従ったシナリオをプロンプトとして記述し，曖昧性の解消を図った．本手法の実現にあたり，取り組むべき課題は以下の 2 点である．

汎用的なシナリオ言語の定義

翻訳エージェントが文脈の不確実さを発見する基準は LLM に大きく依存するため，特定の LLM に依存したシナリオは望ましくない．特に，LLM はプロンプトによって大きく性能に差が生じやすい．したがって，複数の LLM から翻訳エージェントを構築できるように，汎用的なシナリオのシンタックスとセマンティクスの定義が必要である．

シナリオによる翻訳エージェントの制御

翻訳エージェントは，LLM とユーザとのインタラクションを通して翻

訳を確定させていく．このような振る舞いをシナリオとして記述するために，シナリオの語彙を定義し，インタラクションを繰り返しても正確にエージェントを制御できるシナリオを記述する必要がある．

1つ目の課題に対しては、シナリオの「文法」としてシンタックスを、「意味」としてセマンティクスを定義し、それらを用いて操作手順を記述したシナリオを設計した。さらに、YAML のプロンプトと、それと同等の動作を行う自然言語のプロンプトを用いた比較実験を実施した。具体的には、英文ごとにプロンプトを送信し、前の返答を引き継がない形で、多様な英文に対応できるかを検証する「曖昧性検査の網羅性実験」を実施し、さまざまな LLM モデルでプロンプトを実行した際の性能差を評価した。

2つ目の課題に対しては、YAML のプロンプトと、それと同等の動作を行う自然言語のプロンプトを用いて比較実験を実施した。ユーザとのインタラクションを繰り返しても正確にエージェントを制御できるよう、簡潔なシナリオを設計した。具体的には、YAML のシナリオでは、状態を 3 個、イベント (条件) を 2 個、アクション (行動) を 3 個に限定し、簡潔かつ曖昧性のないプロンプトを作成した。また、同じ API に対して連続して翻訳を依頼し、ユーザとのインタラクションを繰り返すことで、シナリオの逸脱頻度、多義語や文化依存の単語の曖昧性検査精度、翻訳精度を分析する「対話継続性の評価実験」を実施した。

大規模言語モデルを用いて評価を行い，提案手法の有効性を検証した．本研究の貢献は以下の通りである．

汎用的なシナリオ言語の定義

「曖昧性検査の網羅性実験」の曖昧性検査精度は、YAML より自然言語のプロンプトの方が GPT-3.5-turbo で 51%、GPT-4o-mini で 38%、GPT-4o で 3%の向上が見られた。一方、翻訳精度は、GPT-3.5-turbo で 58%、GPT-4o-mini で 49%、GPT-4o で 2%の向上が確認された。

シナリオによる翻訳エージェントの制御

「対話継続性の評価実験」のシナリオ逸脱頻度では,GPT-3.5-turbo において自然言語より YAML の逸脱発生率は 60%低かった.GPT-4o-mini では 20%低下したが,GPT-4o では両者の差は見られなかった. 曖昧性検査精度, 翻訳精度ともに,GPT-3.5-turbo では自然言語よりも YAML が 72%低かったのに対し,GPT-4o-mini では 3%高く,GPT-4o では 7%高いという結果が得られた.

Structured Prompts for Translation Agents

Sakie Kaoku

Abstract

With recent globalization, opportunities for communication between different languages and cultures have increased, and the development of machine translation and LLM has lowered the language barrier. However, in sentences containing polysemous or culturally dependent words, traditional machine translation does not consider the context, which can result in inappropriate translations. To solve this problem, 'translation agents' that translate while interacting with the user have gained attention. For example, a system was required that could translate while checking context and nuances like a human translator. To build a translation agent using an LLM, it was necessary to design a scenario for user interaction as a prompt in advance. However, prompts written in natural language tended to be ambiguous, and as dialogue increased, prompts became longer, leading to LLM deviation from the original scenario. To address this, a clearer and more consistent prompt design is required.

Therefore, in this study, we proposed a structured YAML prompt notation for controlling a translation agent. By defining a scenario ' s syntax and semantics, and using a conforming prompt, we aim to eliminate ambiguity. To achieve this, two key issues had to be addressed.

Definition of a multipurpose scenario language

Because the criteria by which a translation agent detects contextual uncertainty depend heavily on the LLM, scenarios that depend on a specific LLM are undesirable. In particular, LLMs exhibited significant performance differences depending on the prompt. Therefore, it is necessary to define the syntax and semantics of a generic scenario so that a translation agent can be constructed from multiple LLMs.

Controlling translation agents by scenarios

The translation agent finalizes the translation through thinking using LLM and interaction with the user. In order to describe such behavior as a scenario, it is necessary to define a vocabulary for scenarios for the translation agent and to describe concise scenarios that can accurately

control the agent even when it repeatedly interacts with the user.

For the first issue, we designed a scenario by defining syntax as its "grammar" and semantics as its "meaning" to structure operational procedures. We then conducted a comparative experiment using YAML and natural language prompts, assessing performance differences across LLM models through a "comprehensiveness experiment for ambiguity testing." This tested the system's ability to handle diverse inputs without carrying over previous responses.

For the second issue, we conducted another comparative experiment to evaluate agent control in repeated user interactions. A concise YAML scenario was designed with three states, two conditions, and three actions for clarity. Through a "dialogue continuity evaluation experiment," we analyzed scenario deviation frequency, ambiguity detection accuracy, and translation accuracy.

We evaluated the effectiveness of the proposed method using a large-scale language model. The contributions of this study are as follows.

Definition of a multipurpose scenario language

The ambiguity detection accuracy in the "comprehensiveness experiment for ambiguity testing" improved with natural language prompts over YAML. GPT-3.5-turbo showed a 51% improvement, GPT-4o-mini 38%, and GPT-4o 3%. Similarly, translation accuracy improved by 58% for GPT-3.5-turbo, 49% for GPT-4o-mini, and 2% for GPT-4o with natural language prompts.

Controlling translation agents by scenarios

In the "dialogue continuity evaluation experiment," scenario deviation frequency was 60% lower with YAML prompts for GPT-3.5-turbo and 20% lower for GPT-4o-mini, while GPT-4o showed no significant difference. Regarding ambiguity detection and translation accuracy, GPT-3.5-turbo performed 72% worse with YAML, whereas GPT-4o-mini performed 3% better, and GPT-4o 7% better with YAML prompts.

翻訳エージェントのための構造化プロンプト

目次

第1章　はじめに

近年, グローバル化の進展により, 異なる言語や文化間でのコミュニケーションを行う機会が増加している. これに伴い, 機械翻訳や大規模言語モデル (Large Language Model, LLM) の技術が発展し, 言語や文化の壁を低くすることが可能になってきた. 一方で, 多義語や文化依存の単語を含む文脈においては, 機械翻訳の技術が正確な意味を解釈することが難しいという課題がある. 多義語とは, 複数の意味を持つ単語を指す. 例えば,「bank」は「銀行」や「川縁」という異なる意味を持つ多義語である. また, 文化依存の単語とは, 文化的な背景によって意味が異なる単語を指す. 例えば,「public school」はアメリカ英語では「公立学校」, イギリス英語では「私立学校」を意味する. このような多義語や文化依存の単語を含む文脈においては, 現在の機械翻訳の技術では正確な意味を判断することが難しい. 例えば, 多義語を含む「I went to the bank.」の文では,「bank」が「銀行」を指すのか「川縁」を指すのか文脈が明確でない限り判断できない. また, 文化依存の単語を含む「I went to the public school.」の文では,「public school」が「公立学校」を指すのか「私立学校」を指すのか文脈が明確でない限り解釈するのが難しい.

このような課題を解決するために, ユーザとのインタラクションを通じて翻訳を生成する「翻訳エージェント」が注目されている. 翻訳エージェントは, 人間の翻訳者が行うように, ユーザに文脈やニュアンスを確認しながら適切な翻訳を行うものである. このような翻訳エージェントを，大規模言語モデル (LLM) を用いて構築するには，ユーザとどのようなインタラクションを行うのかというシナリオを事前にプロンプトとして与える必要がある．しかしながら，自然言語で記述されるプロンプトは曖昧性が高いことに加え，ユーザとのインタラクションが増えるにつれてプロンプトが長くなるため，プロンプトの最初に与えられたシナリオから大規模言語モデルが逸脱する問題がある．

そこで，本研究では，翻訳エージェントを制御するための構造化されたYAMLのプロンプト記法を提案する．具体的には，シナリオのシンタックス (文法) とセマンティクス (意味) を定義するシナリオの仕様と，その言語に従ったシナリオをプロンプトとして記述し，曖昧性の解消を図った．本手法の実現にあたり，取り組むべき課題は以下の2点である．

汎用的なシナリオ言語の定義

翻訳エージェントが文脈やニュアンスの不確実さを発見する基準はLLMに大きく依存するため，特定のLLMに依存したシナリオは望ましくない．特に，LLMはプロンプトによって大きく性能に差が生じやすい．したがって，複数のLLMから翻訳エージェントを構築できるように，汎用的なシナリオのシンタックスとセマンティクスの定義が必要である．

シナリオによる翻訳エージェントの制御

翻訳エージェントは，LLMを用いた思考とユーザとのインタラクションを通して翻訳を確定させていく．このような振る舞いをシナリオとして記述するために，翻訳エージェント用のシナリオの語彙を定義し，ユーザとのインタラクションを繰り返しても正確にエージェントを制御可能な簡潔なシナリオを記述する必要がある．

以下, 本論文では,2章で関連研究のエージェント設計とインタラクション設計言語について説明する. 次に, 第3章で構造化されたYAMLのプロンプトのシナリオの定義について述べる. 続いて4章では, 翻訳エージェントとして制御するシナリオについて述べる. そして5章で3章と4章で説明を行った翻訳エージェントのプロンプトに対する評価を行い,6章にて考察を行う. 最後に,7章にて今後の展望や課題について述べて結論とする.

第2章　関連研究

本章では, 関連研究である, ユーザとのインタラクションを通して翻訳する翻訳エージェントの設計, インタラクション設計言語「Q」,LLM の長文コンテキスト処理について述べる.

2.1　翻訳エージェント

翻訳エージェントの設計に関する研究は, 近年注目されている分野である. 特に,「エージェントメタファー」[1] と呼ばれるアプローチでは, 翻訳を単なる情報伝達として扱うのではなく, ユーザー同士の対話を通じて誤りを修正し, 翻訳の質を向上させる方法が提案されている. この手法では, エージェントを人間の通訳のような役割として設定し, 積極的なインタラクションを通じて翻訳プロセスの効率化を図ることを目的としている. 先行研究 [1] では, ユーザーからのフィードバックを活用し, 翻訳エラーを修正しつつ文脈を考慮した翻訳を生成する仕組みが提案されている. このような修正プロセスは, 本研究が提案する YAML のプロンプトを用いたシナリオ設計においても有用である. 具体的には, エージェントがユーザーとやり取りを行い, 曖昧な表現や文脈の不明瞭さを解消する方法論は, 本研究のシンタックス (文法) およびセマンティクス (意味) の設計に応用可能である. さらに, 先行研究 [1] では, エージェントの自律性とユーザーとの対話を通じて文脈を補足する重要性が強調されている. この点は, 本研究が目指す汎用的なシナリオ記述の設計とも一致しており,YAML プロンプトがエージェントの動作を明確かつ簡潔に制御するための有効な手段であることを示唆している.

以上より, 先行研究は翻訳エージェントの性能向上や評価手法の設計において, 本研究に重要な示唆を与えている. 本研究では, これらの知見を活用し, より高精度で実用的な翻訳エージェントの実現を目指す.

2.2　インタラクション設計言語「Q」

Ishida らによって提案されたインタラクション設計言語「Q」[2][3] は, エージェントと人間のコミュニティ間の柔軟な相互作用を記述するための枠組みを提供している. この言語は, エージェントが修復過程において自律的に意思決定を行う仕組みを構築し, 複雑な文脈に基づいた動的な相互作用を可能にする. 特に, 多義語や文化依存単語といった文脈依存の課題に対しても, ユーザとのイン

タラクションを通じて適切な処理を行う設計思想が本研究の参考となっている.「Q」は, キュー, アクション, ガード付きコマンドを主要要素とし, エージェントの相互作用を効率的に記述する. キューは環境の入力を観察し, アクションは環境を変更する指示を実行, ガード付きコマンドは条件に応じた処理を行う. これにより, 動的で高精度なエージェント設計が可能となる. また,Q は状態遷移を明確に記述できる点でも優れており, 特定の条件に基づいてエージェントが次の状態へ遷移する流れを直感的に表現できる. 例えば, 翻訳エージェントが曖昧な入力を処理する際, ユーザからの補足情報を要求しながら適切な翻訳を導き出すプロセスを設計するのに応用できる.

しかしながら, 本研究では「Q」ではなく「YAML」を採用することを決定した. その理由として,「Q」が Scheme というプログラミング言語を基盤としている点が挙げられる. 一方,YAML は単なるテキストデータとして扱うことが可能であり, プログラムではなくプロンプトとしての記述に親和性が高いと判断した. また,YAML は構造化データを簡潔に表現できるため, 翻訳エージェントのシナリオ記述において,「Q」よりも明確性や簡潔に記述できる.

本研究では,「Q」が示した柔軟な相互作用設計の枠組みを踏襲しつつ,YAML の簡便性とプロンプトへの親和性を活用することで, 翻訳エージェントのシナリオ記述を効率化することを目指している. この選択は, 翻訳エージェントがユーザとのインタラクションを通じて曖昧性を解消し, 文脈を補足するという目標を達成する上で最適な方法であると判断した.

2.3 LLM の長文コンテキスト処理

大規模言語モデル（LLM）は, 長いコンテキストを処理する際に, 提供された情報を無視し, 誤った回答をすることがある. この問題は, プロンプトの設計やモデルの特性に影響されることが指摘されている [4]. さらに,Chain-of-Thought（CoT）推論では, 誤った推論チェーンを生成しながら正しい答えを出すことがあり, 推論過程の忠実性が保証されない [5].LLM の応答の忠実性はプロンプトの記述方法に大きく依存し, 適切な設計によって文脈をより正確に反映できる. また, モデルサイズも影響を及ぼし, 大規模なモデルほどコンテキストへの適応能力は高いが, 同時に事前知識への依存度も増すため, 適切な制御が必要となる [4]. 本研究では,YAML を用いた構造化プロンプトを設計し, ユーザとの長距離インタラクションにおいて LLM の文脈の忠実性を維持する方法を検討する.

第3章　YAMLのシナリオの定義

本章では, 翻訳エージェントを制御するための構造化されたYAMLのプロンプトのシナリオの定義について説明する. まず,YAMLの特徴とJSONとの比較を通じて, その採用理由を明らかにする. 次に,YAMLプロンプトの基本構造であるシンタックス (文法) を解説し, シナリオ内の状態, イベント, アクションの設計方法を示す. また, セマンティクス (意味) について, エージェントがユーザの意図を正確に理解し行動するための定義を述べる.

3.1　YAML

YAML[6] は,「YAML Ain't Markup Language」の再起的頭字語であり, 人間にとって読みやすく使いやすいように設計されたデータシリアライズ形式の言語である. データシリアライズとは, 特定のデータ構造やオブジェクトを, 保存または転送可能な形式に変換するプロセスを指す.YAMLは一般的な日常タスクのために最新のプログラミング言語とうまく動作するように設計されている.YAMLは, 構造化データを簡潔かつ直感的に表現し, 可読性を高めることを目的としている. そのため, 設定ファイルやデータ交換の場面で広く利用されており, シンプルな構文と柔軟性が特徴である. また,JSON(JavaScript Object Notation) と互換性を持ちながら, より直感的で人間に優しい記述を可能にする点で優れている.

YAMLとJSONを比較すると, それぞれの形式に独自の強みがある (図 1) こ

Type	JSON	YAML
Integers	Yes	Yes
Floats	Scientific notation	Scientific notation
Number specifics	Not infinity	Also octal/Hexadecimal
Strings	Yes (Unicode)	Yes (Unicode)
Booleans	Yes	Yes
Arrays	Yes (sequences)	Yes
Associative arrays	Yes (objects)	Yes (mappings)
Null	Yes	Yes
Timestamps	As strings	Yes

Avaibility of basic data types in both formats (naming in parentheses).

図 1: YAMLとJSONの比較
出典 [6]

とがわかる.JSON はシンプルな構文と効率的な処理性能により, リアルタイム性が求められるシステムに適している. 一方で, 構文が固定的でコメントが書けないため, 人間が直接操作する場合には柔軟性が不足することがある. 一方,YAML はインデントによる階層表現とコメント機能により, データの構造を視覚的に理解しやすく, 設定ファイルやドキュメント用途において特に有用である. しかしながら, インデントが不整合になるとエラーを引き起こす可能性があり, 大規模データの処理速度では JSON に劣る場合がある. このように,JSON は機械間の効率的なデータ交換に,YAML は人間が関与するデータ管理や設定用途に最適化されている.

今回の研究において YAML を採用する理由としては, その明確性と汎用性が挙げられる. まず,YAML は構造化された記述形式を持ち, 自然言語による曖昧性を排除することで, 大規模言語モデル (LLM) の制御精度を向上させる効果があると考えられる. また,YAML のプロンプトでは, 状態, イベント (条件), アクション (行動) を簡潔に定義し, 曖昧性のないプロンプトを作成することで, ユーザとの対話を繰り返してもシナリオの一貫性を維持できることが考えられる.YAML はその明確性, 柔軟性, および汎用性により, 翻訳エージェントの制御や大規模言語モデルの応答精度向上において重要な役割を果たす. 以上の理由から, 本研究では YAML を採用した.

3.2 シンタックス (文法) の定義

シンタックス [7] とは, 言語において単語や句を適切に並べ, 文法的に正しい文を構成するための規則を指す. これらのルールは, 言語の文法構造を定義できる.YAML プロンプト記法におけるシンタックスは, エージェントの振る舞いを記述するための文法規則を明確に定義するものである. 以下に, シンタックス設計の基本的な構造とその特徴を示す. 文法的な正確性を保つことは, 汎用的な YAML 言語のプロンプトを作成するために重要である. シンタックスは, シナリオ (Scenario), アクション (Action), イベント (Event) の 3 個作成した. 以下に, それぞれのシンタックス設計の基本的な構造とその特徴を示す.

3.2.1 シナリオ (Scenario)

まず, シナリオ (Scenario) とは, 特定の目的を達成するために与える指示の流れである. ここでは, そのシナリオ全体の名前や, エージェントの振る舞いを記述するルールを以下のように定義した.

- **名前 (name)**: シナリオ全体の名前を定義する.
- **状態 (States)**: シナリオ内でのエージェントの異なる状態をリストにする. 各状態には以下の要素が含まれる:
  - **名前 (name)**: 状態の名前を定義する.
  - **ルール (rules)**: 状態遷移の条件を記述する.
    - **イベント (event)**: 状態遷移を引き起こすトリガーを定義する.
    - **アクションズ (actions)**: イベントが発生した際に実行される動作をリストにする.
      - **アクション (action)**: イベントが発生した際に実行される動作を指定する.
    - **遷移先 (go_state)**: イベント発生後に移行する次の状態を指定する.

実際のシナリオ (Scenario) のシンタックスは, 図 2 である.

3.2.2 アクション (Action)

アクション (Action) とは, シナリオ内でエージェントが状態遷移を引き起こすトリガーとなるイベントに応じて実行する具体的な動作である. 以下のように定義を行った.

- **名前 (name)**: アクションの名前を定義する.
- **機能 (function)**: アクションが実行する具体的な処理の内容を定義する.

実際のアクション (Action) ののシンタックスの構成要素は, 図 3 である.

```
Below is the syntax.
Please understand the syntax. I'll give you the scenario later, so please act accordingly.
def_scenario:
 name: "Specifies the name of the scenario."
 states: "Lists the different states within the scenario."
  name: "Specifies the name of the state."
  rules: "Lists the rules governing transitions between states."
   event: "Specifies the event that triggers the transition."
   actions: "Lists the actions that occur when the event is triggered."
    action: "Specifies the name of the action to be executed."
   go_state: "Specifies the next state the agent transitions to after the event occurs."
```

図 2: シナリオ (Scenario) のシンタックス

```
def_action:
 name: "Specifies the name of the action."
 function: "Describes the functionality of the action."
```

図 3: アクション (Action) のシンタックス

```
def_event:
 name: "Specifies the name of the event."
 content: "Describes the conditions or triggers for the event."
```

図 4: イベント (Event) のシンタックス

3.2.3 イベント (Event)

イベント (Event) とは, 状態遷移を引き起こす条件やトリガーである. 以下のように定義を行った.

- **名前 (name)**: イベントの名前を定義する.
- **内容 (content)**: 状態遷移を引き起こす具体的な条件を定義する.

実際のイベント (Event) ののシンタックスの構成要素は, 図 4 である.

3.3 セマンティクス (意味) の定義

セマンティクス (Semantics)[7] とは, シナリオにおける各要素の意味や動作を定義するものであり, 単語やフレーズがさまざまな文脈でどのように意味を伝えるかに焦点を当てている. イベントが発生した際に実行されるアクションの内容や, 状態遷移後のエージェントの振る舞いを明確に記述するための基盤となるものである. 以下のようにアクション, イベントのセマンティクスをそれぞれ定義した.

3.3.1 アクション (Action)

アクションは, イベントが発生した際に実行される動作である. 多義語や文化依存の単語をユーザとのインタラクションを通じて翻訳を生成する「翻訳エージェント」を作成するにあたり,3 個のアクションを作成した. 具体的には,「曖昧性を確認」,「説明を求める」,「翻訳と表示」の 3 個のアクションを作成した. 以下のようにそれぞれ定義した.

- **check_ambiguity**: ChatGPT 自体が, 受信したメッセージが語義の曖昧さであるかどうかを判断する.

- **ask_clarification**: メッセージ内の語義の曖昧さを明確にするには, ユーザーに英語で具体的な質問をする.
- **translate_and_display**: メッセージを日本語に翻訳し, 翻訳結果を表示する.

実際のアクション (Action) のセマンティクスは, 図 5 である.

3.3.2 イベント (Event)

イベントは, 状態遷移を引き起こす条件である. 多義語や文化依存の単語をユーザとのインタラクションを通じて翻訳を生成する「翻訳エージェント」を作成するにあたり,2 個のアクションを作成した. 具体的には,「曖昧性を確認」,「説明を求める」,「翻訳と表示」の 3 個を作成した. 以下のようにそれぞれ定義した.

- **receive_message**: ユーザーからメッセージを受信したときに発生する.
- **ambiguity_detected**: 受信したメッセージで語義の曖昧さが検出されたときに発生する.

実際のアクション (Action) のセマンティクスは, 図 6 である.

```
def_action:
 name: check_ambiguity
 function: "ChatGPT itself will determine whether the received message is a word sense ambiguity."

def_action:
 name: ask_clarification
 function: "To clarify any word sense ambiguity in your message, ask the user a specific question in English."

def_action:
 name: translate_and_display
 function: "Translates the message into Japanese and displays the translated result."
```

図 5: アクション (Action) のセマンティクス

```
def_event:
 name: receive_message
 content: "This event occurs when a message is received from the user."

def_event:
 name: ambiguity_detected
 content: "This event is triggered when word sense ambiguity is detected in the received message."
```

図 6: イベント (Event) のセマンティクス

第 4 章　翻訳エージェントのシナリオ

本章では, 翻訳エージェントを制御するための汎用的なシナリオの全体構造について述べる. 第 3 章で記述した各要素 (シナリオ, アクション, イベント) の具体的なシンタックスやセマンティクスを基盤として, 本章ではこれらを組み合わせたシナリオ全体の設計ついて説明する.

4.1　汎用的なシナリオの設計方針

汎用的なシナリオを設計する際, 明確で簡潔な構造になっているか注意した. 状態やイベントやアクションが増えると, シナリオが複雑化しやすいので, 対策として状態やイベント, アクションの数を必要最低限に絞り, 簡潔な設計を心がけた. また, 曖昧なプロンプトはエージェントの誤動作を引き起こす可能性があるため, 対策として明確なルールを設け, 適切な処理を定義した. さらに, 汎用的な設計を心がけ, さまざまな状況で再利用可能なシナリオを作成した. 大規模言語モデルの種類によってシナリオの挙動が異なる場合がある問題を解決するために,YAML のような構造化された記述を採用し, モデル間の一貫性を保った. これらを実現することで, 翻訳エージェントの効率的かつ信頼性の高い動作が可能となった.

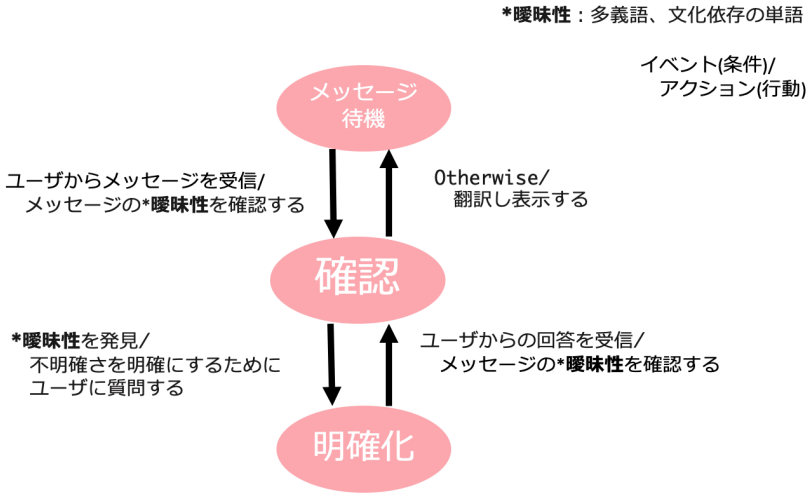

図 7: シナリオ全体の状態遷移図

4.2　シナリオの全体設計

汎用的なシナリオの全体の状態遷移図は, 図 7 のように設計される. 状態は, 「メッセージ待機」,「確認」,「明確化」の 3 個で構成した. セマンティクスで定義した通り, イベントは 2 個使用し, アクションは 3 個使用した. この状態遷移図の「/」より前が条件であるイベントでそのイベントが起こった場合, 動作であるアクションを行うように記述している. また, ここでの曖昧性とは, 文脈やニュアンスを確認しながら翻訳をしないと誤った翻訳になる可能性がある, 多義語や文化依存の単語を含むものである. 以下でそれぞれの状態の設計についても述べる.

4.2.1　「メッセージ待機」状態

「メッセージ待機」状態の状態遷移図は, 図 8 のように設計される.「メッセージ待機」状態から「確認」状態へ移動するときのイベントは, ユーザからメッセージを受信するであり, このイベントが起こるとメッセージの曖昧性を確認するアクションを行う. この「メッセージ待機」状態の状態遷移図を YAML にし

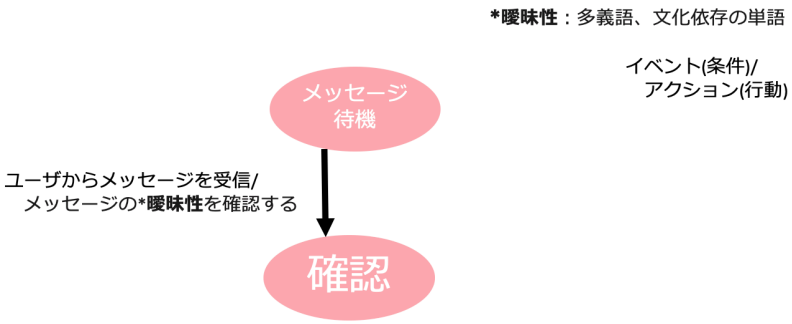

図 8:「メッセージ待機」状態の遷移図

```
def_scenario:
  name: "Message Clarification and Translation"
  states:
    - name: "Waiting for Message"
      rules:
        - event: "receive_message"
          actions:
            - action: "check_ambiguity"
          go_state: "Confirmation"
```

図 9:「メッセージ待機」状態の YAML

たものが図 9 である.

4.2.2 「確認」状態

「確認」状態の状態遷移図は, 図 10 のように設計される.「確認」状態から「明確化」状態へ移動するときのイベントは, 曖昧性を発見するであり, このイベントが起こると曖昧性を明確にするためにユーザに質問するアクションを行う.「確認」状態から「メッセージ待機」状態へ移動するときのイベントは, 曖昧性を発見しないであり, このイベントが起こると翻訳し表示するアクションを行う. この「確認」状態の状態遷移図を YAML にしたものが図 11 である.

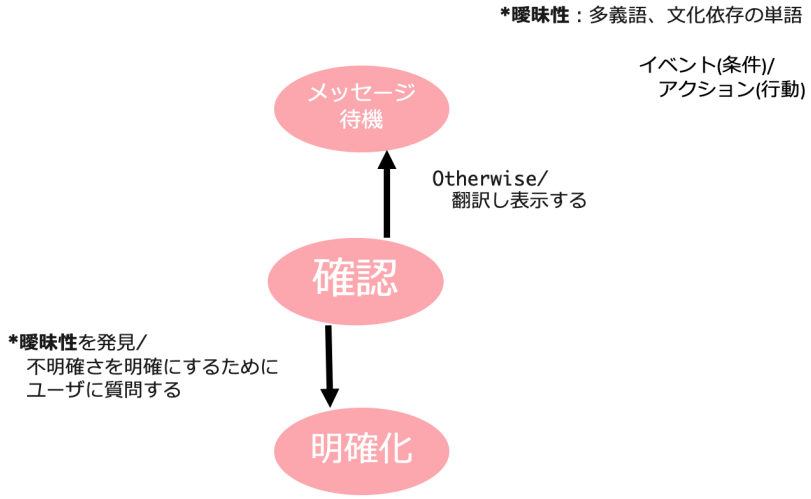

図 10: 「確認」状態の遷移図

```
- name: "Confirmation"
  rules:
    - event: "ambiguity_detected"
      actions:
        - action: "ask_clarification"
      go_state: "Clarification"
    - event: "otherwise"
      actions:
        - action: "translate_and_display"
      go_state: "Waiting for Message"
```

図 11: 「確認」状態の YAML

4.2.3 「明確化」状態

「明確化」状態の状態遷移図は, 図 12 のように設計される.「明確化」状態から「確認」状態へ移動するときのイベントは, ユーザからの回答を受信するであり, このイベントが起こるとメッセージの曖昧性を確認するアクションを行う. この「確認」状態の状態遷移図を YAML にしたものが図 13 である.

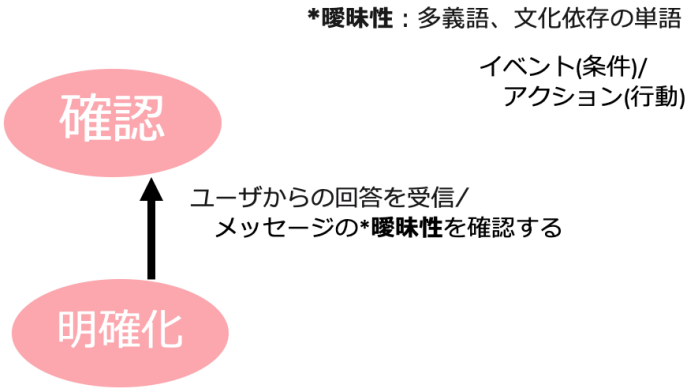

図 12: 「明確化」状態の遷移図

```
- name: "Clarification"
  rules:
    - event: "receive_message"
      actions:
        - action: "check_ambiguity"
      go_state: "Confirmation"
```

図 13: 「明確化」状態の YAML

第5章　評価

本章では, 文脈を正確に反映できる翻訳エージェントの実現を目的とし, 特定の LLM に依存しない構造化されたプロンプト記法の有効性を検証するために第 3 章,4 章で説明した YAML のプロンプトと自然言語のプロンプトの比較実験について述べる. 初めに, 評価指標について述べる. 次に, 評価手法について述べ, 最後に結果を述べる.

5.1　評価指標

翻訳エージェントの YAML, 自然言語のプロンプトの性能を評価するために設定した評価項目について述べる. 評価は, エージェントのシナリオ逸脱頻度と多義語や文化依存の単語の曖昧性の検査精度と翻訳精度の 3 つの観点から行う. これにより, 提案する YAML プロンプトの有効性を検証し, エージェントがユーザとのインタラクションを通じて適切に動作するかを確認する. 以下に, それぞれの評価項目について詳細を示す.

5.1.1　シナリオ逸脱頻度

シナリオ逸脱とは, 翻訳エージェントが事前に定義したシナリオから逸脱し予期しない動作をする回数を測定する評価指標である. 具体的には, ユーザとのインタラクションを繰り返す中で, エージェントがシナリオから外れた場合の回数を測定する.

5.1.2　曖昧性の検査精度

曖昧性の検査精度とは, 翻訳エージェントが文脈によって意味が変化する多義語や文化依存の単語を正しく検出し, 適切に判断できる能力を測定する評価指標である. 具体的には, 曖昧な表現を含むテスト文を使用し, エージェントが曖昧性を正確に判定し, 適切な質問を通じてユーザから必要な追加情報を引き出す能力を検証する. したがって, 翻訳エージェントが曖昧性がある文に対して曖昧性を正しく判断し, 聞き返した数が多い方が曖昧性の検査精度が高いことになる. 曖昧性検査精度は, 曖昧性がある英文の中で何文曖昧性を判断し聞き返すことができたかを%で記述する.

5.1.3　翻訳精度

翻訳精度とは, 翻訳エージェントが与えられた文脈に基づいて, 正確で適切な翻訳を生成できる能力を測定する評価指標である. この評価では, 多義語や文化

依存の単語を含むテスト文を用い, エージェントがこれらの翻訳をどれだけ正確に処理できるかを検証する. さらに, 翻訳精度の評価においては, 曖昧性のある文章に対し, エージェントが適切に聞き返しを行わなかった場合, その翻訳は正しく翻訳を行ったとみなさないものとする. つまり, 曖昧性を含む文を適切に処理するためには, まずエージェントが曖昧性を検出し, 必要な追加情報を取得した上で翻訳を行うことが求められる. 翻訳精度は, 全文のうち何文翻訳を正確に出力できたかを%で記述する.

5.2 評価手法

5.2.1 ベースライン

比較対象である YAML のプロンプトと自然言語のプロンプトについて述べる.YAML のプロンプトは, 第 3,4 章のように構造が明確になるよう記述され, 各イベントやアクション, 状態遷移が階層的に記述している. 自然言語のプロンプトでは, そのYAMLの構造的に書いたプロンプトをもとに, 文章に変換した. YAML を基にした自然言語プロンプトは図 14 になる.

このように, 自然言語のプロンプトでは,YAML で明確に定義されていた状態やルールや, 形式的な要素を省略して, 利用者が動作を直感的に理解できる文書の記述に変換した.

5.2.2 LLM の種類

本研究では,YAML プロンプトと自然言語プロンプトの性能を比較するために, 以下の大規模言語モデルを使用した. 各モデルは異なる規模や設計を持つ

```
You clarify and translate the message.
Please act as follows:
When receiving a message from a user, ChatGPT itself will determine whether
the received message is a word sense ambiguity.
If a word sense ambiguity is detected in the received message,
ask the user a specific question in English to clarify any word sense ambiguity in your message.
After receiving clarification from a user, ChatGPT itself will determine
whether the received message is a word sense ambiguity.
And if a word sense ambiguity is detected in the received message,
ask the user a specific question in English to clarify any word sense ambiguity in your message.
If no a word sense ambiguity is detected in the received message,
translates the message into Japanese, displays the translated result and
wait to receive the next message from the user.
```

図 14: 自然言語のプロンプト

が,YAML プロンプトと自然言語プロンプトがモデルに左右されずに汎用的に翻訳エージェントを動かすことができるかを検証した. 大規模言語モデルの GPT のモデルでは, オープン AI が提供する最先端の自然言語処理モデルであり, 多様なタスクにおいて高い汎用性を持つ. 本研究では, 以下の 3 個のモデルを採用した：

- **GPT-3.5-turbo**: 軽量で応答速度が高く, 幅広いタスクに対応可能なモデル.
- **GPT-4o-mini**: GPT-4 の軽量版で, 計算リソースを抑えつつ精度を確保したモデル
- **GPT-4o**: GPT-4 をベースとした高精度モデルで, 複雑なタスク処理に適しているモデル

5.2.3 データセット

比較実験には, 多義語や文化依存単語を含むテスト文と含まないテスト文を用いた. 曖昧性を含む文章と含まない文章を組み合わせた理由は以下で述べる. 比較実験では, 多義語や文化依存単語を含むテスト文と含まないテスト文を用いた. これは, 現実の翻訳タスクでは曖昧性を含む文章と明確な文章が混在しているため, エージェントの性能をより実際の利用環境に近い条件で評価し, 曖昧性を解消するプロンプトの効果を検証するためである. 特に, 曖昧性を含む文章では, 多義語や文化依存単語を適切に判定し, ユーザから必要な情報を引き出す能力が求められる. 一方で, 曖昧性を含まない文章では, 余計な確認を行わずに正確で効率的な翻訳を生成できるかが評価の対象となる. さらに, 曖昧性の有無によるエージェントの挙動を比較することで, 曖昧性を解消するシナリオの有効性を詳細に評価し, その適用範囲を明確にすることができる. このようにして, 多義語や文化依存単語を含む曖昧な文章と, 曖昧性を含まない明確な文章を組み合わせることで, エージェントの実用性, 効率性, シナリオの逸脱頻度, 翻訳精度を評価した.

全体テスト文の総数は 200 文である. その内訳は以下の通りである.

- **語句の曖昧性あり**：100 文
  - **多義語**：60 文
  - **文化依存の単語**：20 文
  - **口語表現の単語**：20 文
- **語句の曖昧性あり**: 100 文

なお, これらのデータセットは実験ごとに使用される文数が異なる. 詳細については, 次節の 5.2.4 評価実験の概要で説明する.

5.2.4　実験手順

本研究では,YAML のプロンプトと自然言語のプロンプトの比較実験を行い, ユーザとのインタラクションを通じてエージェントを正確に制御可能なシナリオ設計の有効性を検証した.2 個の異なる実験を実施し, それぞれ異なるデータセットを用いた. 以下で具体的に説明する.

1 つ目「曖昧性検査の網羅性実験」では, さまざまな英文に対して翻訳エージェントが多義語や文化依存の単語の曖昧性検査精度, 翻訳精度の分析を行った. テスト文は, 合計 200 文を使用した. 詳細は, 曖昧性あり 100 文 (多義語:60 文, 文化依存の単語：20 文, 口語表現の単語：20 文), 曖昧性なし 100 文である. 曖昧性検査の網羅性実験では, プロンプトを英文を送るごとに一緒に送り, その前の返答分は受け継がないものとし, さまざまな英文に対応できるかといった網羅性を検証した.

2 つ目の「対話継続性の評価実験」では, 同じ API に翻訳を続けて依頼したときユーザとのインタラクションを繰り返すことで，シナリオの逸脱頻度, 多義語や文化依存の単語の曖昧性の検査精度, 翻訳精度を分析した. テスト文は合計 100 文 (曖昧性あり 50 文, 曖昧性なし 50 文) を使用した. 詳細は, 曖昧性あり 50 文 (多義語:30 文, 文化依存の単語：10 文, 口語表現の単語：10 文), 曖昧性なし 50 文である. 対話継続性の評価実験では, 連続した対話の中でのエージェントの継続性を評価した.

5.3　結果

5.3.1　曖昧性検査の網羅性実験

曖昧性検査の網羅性実験では,GPT-3.5-turbo,GPT-4o-mini,GPT-4o のそれぞれのモデルにおいて表 1,2,3 より YAML よりも自然言語の方が曖昧性の検査精度と翻訳精度が高いことが確認された. 曖昧性検査精度は, 曖昧性がある英文の 100 文の中で何文曖昧性を判断し聞き返すことができたかを%で記述し, 翻訳精度は, 全文 200 文のうち何文翻訳を正確に出力できたかを%で記述した. 翻訳精度の評価においては, 曖昧性のある文章に対し, エージェントが適切に聞き返しを行わなかった場合, その翻訳は正しく翻訳を行ったとみなさないものとする. 以下に, 各モデルの結果を述べる.

GPT-3.5-turbo モデル　表 1 で示す通り,GPT-3.5-turbo モデルでは,YAML プロンプトに比べて自然言語プロンプトの方が, 曖昧性の検査精度および翻訳精度が高い結果が得られた. 具体的には, 曖昧性の検査精度は YAML プロンプトでは 7%であったのに対し, 自然言語プロンプトでは 58%と大幅に向上した. また, 翻訳精度に関しても,YAML プロンプトが 10%であったのに対し, 自然言語プロンプトでは 68%と差が見られた. この結果から,GPT-3.5-turbo モデルでは,YAML プロンプトよりも自然言語プロンプトの方が, 曖昧な表現を適切に認識し, より正確な翻訳を生成できることが示唆された. また, 曖昧性の検出に関するプロセスに違いが見られた. 曖昧性のある英文を送信した際, 自然言語プロンプトを用いた場合は, すぐに曖昧性を聞き返す質問が行われ, スムーズに処理が進んだ. 一方,YAML プロンプトを用いた場合, 特に GPT-3.5-turbo のようなモデルでは,「Let me check for any word sense ambiguity. Please hold on a moment.」 のような応答を返した後に曖昧性の聞き返しを行う傾向が見られ 1 回目から曖昧性を判断し聞き返すことができていなかった. さらにプロンプトの長さに関しては,YAML プロンプト (1372 トークン) は自然言語プロンプト (264 トークン) よりも約 5 倍長いことが確認され, プロンプトの長さが短い自然言語プロンプトの

表 1: GPT-3.5-turbo モデルでの曖昧性検査精度と翻訳精度

	平均プロンプト長 (トークン数)	**曖昧性検査精度 (%)**	**翻訳精度 (%)**
YAML	1372	7	10
自然言語	264	58	68

表 2: GPT-4o-mini モデルでの曖昧性検査精度と翻訳精度

	平均プロンプト長 (トークン数)	**曖昧性検査精度 (%)**	**翻訳精度 (%)**
YAML	1195	53	36
自然言語	328	91	85

表 3: GPT-4o モデルでの曖昧性検査精度と翻訳精度

	平均プロンプト長 (トークン数)	**曖昧性検査精度 (%)**	**翻訳精度 (%)**
YAML	921	95	94
自然言語	352	98	96

方が, 曖昧性検査精度および翻訳精度の両方において高い値を示した.

GPT-4o-mini モデル　表 2 に示すように,GPT-4o-mini モデルにおいても, 自然言語プロンプトの方が YAML プロンプトよりも曖昧性検査精度および翻訳精度において優位性が見られた. 具体的には, 曖昧性検査精度は YAML プロンプトが 53%であったのに対し, 自然言語プロンプトでは 91%と向上した. また, 翻訳精度に関しても,YAML プロンプトでは 36%であったが, 自然言語プロンプトでは 85%となった. プロンプトの長さに関しては,YAML プロンプト (1195 トークン) は自然言語プロンプト (328 トークン) よりも約 3.6 倍長いことが確認され, プロンプトの長さが短い自然言語プロンプトの方が, 曖昧性検査精度および翻訳精度の両方において高い値を示した.

GPT-4o モデル　表 3 に示す通り,GPT-4o モデルはすべての評価項目において最も高い精度を示した. 曖昧性検査精度は YAML プロンプトで 95%, 自然言語プロンプトで 98% となり, どちらのプロンプト形式でも非常に高い曖昧性検出能力が確認された. また, 翻訳精度に関しても,YAML プロンプトが 94%, 自然言語プロンプトが 96%となり, 高い翻訳精度が確認された. 曖昧性のある英文を送信した際,GPT-4o モデルでは,YAML プロンプトと自然言語プロンプトのいずれを用いた場合でも, 即座に適切な曖昧性の聞き返しが行われる傾向が見られた.GPT-4o モデルでは,YAML プロンプトと自然言語プロンプトのいずれを用いた場合でも, 曖昧性のある英文に対して即座に聞き返しを行う傾向が見られた. さらに, 曖昧性の検出までに要する時間は他のモデルと比べて差は小さかった. プロンプトの長さに関しては,YAML プロンプト (921 トークン) は自然言語プロンプト (352 トークン) よりも約 2.6 倍長いことが確認された. しかしながら, 他のモデルと比較して,GPT-4o では YAML プロンプトと自然言語プロンプトの性能差が小さく, どちらのプロンプトを用いても高い精度を維持できる結果となった.

GPT-3.5-turbo モデルでは,YAML プロンプトの曖昧性検査精度および翻訳精度が極めて低く, 自然言語プロンプトの方が大幅に優れた結果を示した.GPT-4o-mini および GPT-4o モデルでは,YAML プロンプトの方が高い曖昧性検査精度および翻訳精度を示した. 一方, 自然言語プロンプトはシナリオの逸脱を防ぎつつ, 比較的高い精度を維持することができた. 特に GPT-4o モデルでは, いずれのプロンプト形式でもシナリオの逸脱が発生せず, 継続的な対話の精度が向上していることが確認された.

5.3.2 対話継続性の評価実験

対話継続性の評価実験では,GPT-3.5-turbo,GPT-4o-mini,GPT-4o のそれぞれのモデルにおいて, 表 4,5,6 より,YAML プロンプトと自然言語プロンプトのシナリオ逸脱頻度, 曖昧性検査精度, および翻訳精度を比較した. シナリオの逸脱に関しては YAML よりも自然言語の方が優位性が確認できた. しかしながら,GPT-4o-mini,GPT-4o のそれぞれのモデルにおいて自然言語よりも YAML の方が曖昧性の検査精度と翻訳精度が高いことが確認された. シナリオ逸脱頻度は, エージェントがシナリオの流れから逸脱した回数を記録し, 曖昧性検査精度は, シナリオを逸脱しなかった回で曖昧性がない英文 50 文に対して聞き返さなかったか, 曖昧性がある英文 50 文のうち何文が適切に曖昧性を判断し聞き返すことができたかを%で示した. 翻訳精度は, 全文 100 文のうち何文が正確に翻訳を出力できたかを%で記述した. 翻訳精度の評価においては, 曖昧性のある文章に対し, エージェントが適切に聞き返しを行わなかった場合, その翻訳は正しく翻訳を行ったとみなさないものとする. 以下に, 各モデルの結果を述べる.

表 4: GPT-3.5-turbo モデルでのシナリオの逸脱, 曖昧性検査精度, 翻訳精度

	トークン数	**逸脱**	**曖昧性検査精度 (%)**	**翻訳精度 (%)**
YAML	0	5	0	0
自然言語	5574	2	72	72

表 5: GPT-4o-mini モデルでのシナリオ逸脱, 曖昧性検査精度, 翻訳精度

	トークン数	**逸脱**	**曖昧性検査精度 (%)**	**翻訳精度 (%)**
YAML	14106	3	88	88
自然言語	8144	2	85	85

表 6: GPT-4o モデルでのシナリオ逸脱, 曖昧性検査精度, 翻訳精度

	トークン数	**逸脱**	**曖昧性検査精度 (%)**	**翻訳精度 (%)**
YAML	8897	0	92	92
自然言語	4060	0	85	85

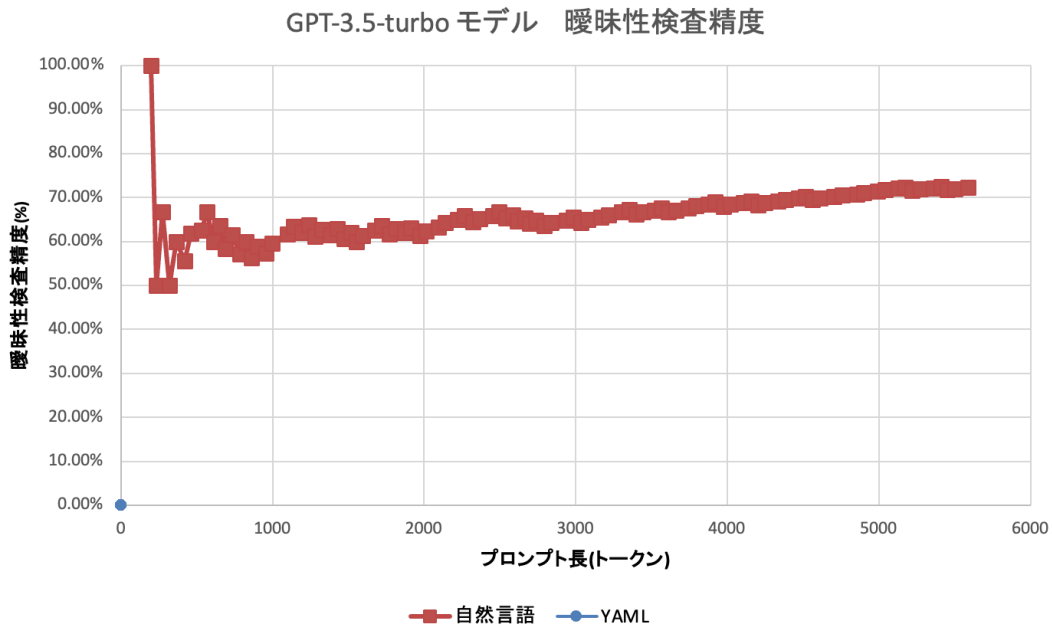

図 15: GPT-3.5-turbo モデルでのプロンプト長と曖昧性の検査精度

GPT-3.5-turbo モデル　表 4 に示すように,GPT-3.5-turbo モデルでは,YAML プロンプトを用いた場合, シナリオ逸脱頻度は 5 回とも全て逸脱しており, 全て逸脱しているためトークン数, 曖昧性検査精度および翻訳精度は 0%とした. 一方, 自然言語プロンプトを用いた場合, 逸脱頻度は 2 回に減少し, 逸脱しなかった 3 回の中で曖昧性検査精度, 翻訳精度ともに 71%と向上した. プロンプトの長さについては, 自然言語プロンプトは 4240 トークンであった.

図 15 は,GPT-3.5-turbo モデルで実施したテストにおいて, 合計 100 文を 5 回繰り返したうち, 逸脱しなかったケースにおけるプロンプトの長さと曖昧性検査精度を示している.YAML プロンプトは 5 回全てシナリオを逸脱したため 0 となったが, 自然言語プロンプトはプロンプト長が増加しても 65%前後に留まる傾向が見られた.

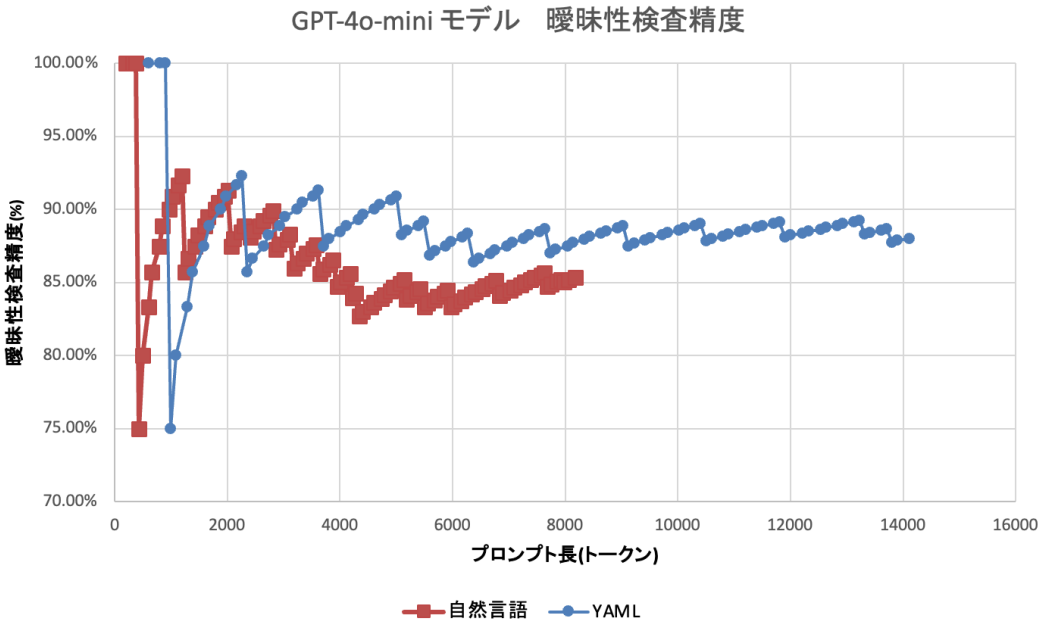

図 16: GPT-4o-mini モデルでのプロンプト長と曖昧性の検査精度

GPT-4o-mini モデル　表 5 に示す通り,GPT-4o-mini モデルでは, シナリオ逸脱頻度が YAML プロンプトで 3 回, 自然言語プロンプトで 2 回起こった. 曖昧性検査精度と翻訳精度は, 逸脱しなかった YAML プロンプトの 2 回, 自然言語プロンプトの 3 回で算出した. 曖昧性検査精度と翻訳精度ともに,YAML プロンプトで 88%, 自然言語プロンプトで 85%となり,YAML プロンプトの方が高い精度を示し優位であることが確認された. プロンプトの長さに関しては,YAML プロンプトのトークン数が 14106 トークンと非常に長かったのに対し, 自然言語プロンプトが 8144 トークンであった.

図 16 は,GPT-4o-mini モデルで実施したテストにおいて, 合計 100 文を 5 回繰り返したうち, 逸脱しなかったケースにおけるプロンプトの長さと曖昧性検査精度を示している.YAML プロンプトはプロンプト長が増加しても曖昧性検査精度が 85%以上を維持し, 安定した結果を示した. 一方, 自然言語プロンプトは 80%前後に留まる傾向が見られた.

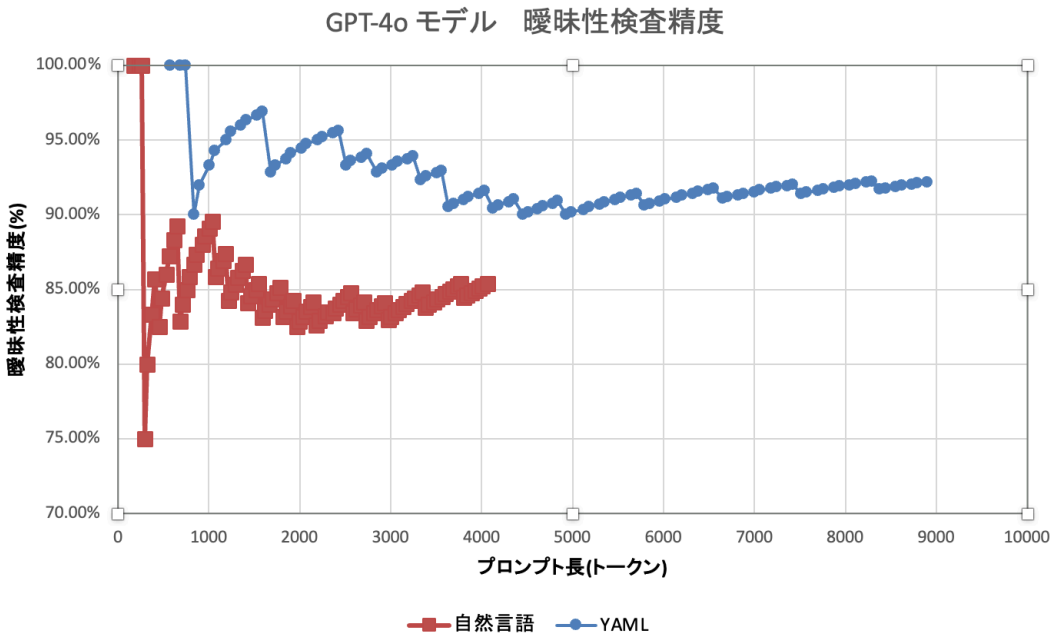

図 17: GPT-4o モデルでのプロンプト長と曖昧性の検査精度

GPT-4o モデル　表 6 に示す通り,GPT-4o モデルでは,YAML プロンプトと自然言語プロンプトのいずれを用いた場合でも, シナリオ逸脱頻度は 0 回と安定した対話が維持された. 曖昧性検査精度は YAML プロンプトで 92%, 自然言語プロンプトで 85%となり,YAML プロンプトの方が高い精度を示した. また, 翻訳精度は YAML プロンプトで 92%, 自然言語プロンプトで 85%となり,YAML プロンプトの方が優位であることが確認された. プロンプトの長さに関しては,YAML プロンプトのトークン数が 8897 トークン, 自然言語プロンプトが 4060 トークンであった.

図 17 は,GPT-4o モデルで実施したテストにおいて, 合計 100 文を 5 回繰り返したうち, 逸脱しなかったケースにおけるプロンプトの長さと曖昧性検査精度を示している.YAML プロンプトはプロンプト長が増加しても曖昧性検査精度が 90%以上を維持し, 安定した結果を示した. 一方, 自然言語プロンプトは 85%前後に留まる傾向が見られた.

第6章　考察

本章では,5 章の YAML のプロンプトと自然言語のプロンプトの比較実験の結果より曖昧性検査の網羅性対話継続性の評価実験の考察について述べる. 具体的には, シナリオの逸脱頻度, 曖昧性の検査精度, 翻訳精度について述べる.

6.1　シナリオの逸脱頻度

曖昧性検査の網羅性実験は, プロンプトを英文を送るごとに一緒に送り, その前の返答分は受け継がないものとし, さまざまな英文に対応できるかといった網羅性を検証したためシナリオの逸脱頻度は測定しなかった。

対話継続性の評価実験では,GPT-3.5-turbo において自然言語より YAML の逸脱発生率は 60%低かった.GPT-4o-mini では 20%低下したが,GPT-4o では両者の差は見られなかった. したがって、低精度なモデルでは自然言語プロンプトの方がシナリオ逸脱を抑えやすく, 高精度なモデルではシナリオの逸脱という点に関しては YAML, 自然言語ともに差がないことがわかる. 低精度なモデルにおいて YAML プロンプトのシナリオ逸脱が増えた原因としては曖昧性検査の網羅性実験の YAML の曖昧性検査精度が自然言語よりも GPT-3.5-turbo では 51%、GPT-4o-mini では 38%低いためだと考えられる. また、曖昧性検査の網羅性実験の YAML の翻訳精度が自然言語よりも GPT-3.5-turbo では 58%、GPT-4o-mini では 49%低いためだと考えられる. これは、YAML のシナリオの逸脱が起こりやすかったのは 1 行目から逸脱していることが多くこれは YAML で 1 文ずつで翻訳すること自体ができていないからである.

6.2　曖昧性の検査精度

曖昧性検査の網羅性実験の曖昧性検査精度では,YAML のプロンプトより自然言語のプロンプトの方が GPT-3.5-turbo では 51%, 高く,GPT-4o-mini では 38%高く,GPT-4o では 3%高かった. この結果から、自然言語のプロンプトは全モデルにおいて優位性を示した。これは YAML のプロンプトの長さが長く低精度なモデルにおいて理解が難しく, それに対し自然言語のプロンプトの長さが短くどのモデルにおいても理解しやすいためと考えられる. モデルごとの性能差に着目すると,GPT-4o モデルは曖昧性検出精度において YAML95%, 自然言語 98%で、最も優れていた。GPT-4o-mini モデルは GPT-4o モデルと比べて YAML42%低

く, 自然言語 7%低かった。また、GPT-3.5-turbo では GPT-4o モデルと比べて YAML88%低く, 自然言語 40%低かった. このことから, モデルの性能向上が曖昧性処理能力の向上に寄与していることが示される.GPT-3.5-turbo モデルでは曖昧性検査精度が GPT-4o モデルと比べて YAML88%低く, 自然言語 40%低かったことから曖昧性処理の能力に課題が見られ, 特に YAML を使用した場合に検出漏れが多い傾向が確認できた. 低精度モデルにおいては,YAML の構造化された記述を適切に解釈する能力が十分でなく, 曖昧性の判断や適切な聞き返しが行われない場合が多かったと考えられる. したがって, 現在の YAML のプロンプトでは構造化しすぎており, モデルの精度が低いのに対し, 曖昧性を聞き返すまで到達していないことがわかる.

対話継続性の評価実験では,GPT-3.5-turbo では自然言語よりも YAML が 72%低かったのに対し,GPT-4o-mini では 3%高く,GPT-4o では 7%高いという結果が得られた. したがって, 高精度なモデルにおいて YAML の優位性が確認された.YAML のプロンプトは長くなっても構造化されているため, 一貫して曖昧性を判断し聞き返すことができた. 一方で, 自然言語のプロンプトは長くなるほど曖昧性の判断が難しくなり, 聞き返しの精度が低下した.

6.3 翻訳精度

翻訳精度に関しても, 曖昧性検査精度と同様の傾向が確認された.

曖昧性検査の網羅性実験の翻訳精度では,YAML のプロンプトより自然言語のプロンプトの方が GPT-3.5-turbo では 58%, 高く,GPT-4o-mini では 49%高く,GPT-4o では 2%高かった. この結果から、自然言語のプロンプトは全モデルにおいて優位性を示した。これは YAML のプロンプトの長さが長く低精度なモデルにおいて理解が難しく, それに対し自然言語のプロンプトの長さが短くどのモデルにおいても理解しやすいためと考えられる. モデルごとの性能差に着目すると,GPT-4o モデルは翻訳精度において YAML94%, 自然言語 96%で、最も優れていた。GPT-4o-mini モデルは GPT-4o モデルと比べて YAML58%低く, 自然言語 11%低かった。また、GPT-3.5-turbo では GPT-4o モデルと比べて YAML84%低く, 自然言語 28%低かった. このことから, モデルの性能向上が翻訳精度の能力の向上に寄与していることが示される.GPT-3.5-turbo モデルでは曖昧性検査精度が GPT-4o モデルと比べて YAML84%低く, 自然言語 28%低かったことから誤訳や翻訳の抜けが発生しやすい傾向が見られた。特に YAML を使用

した場合に検出漏れが多い傾向が確認できた. 低精度モデルにおいては,YAML の構造化された記述を適切に解釈する能力が十分でなく, 翻訳が行われない場合が多かったと考えられる. 一方で, 自然言語のプロンプトは短く, どのモデルでも比較的理解しやすいため, 翻訳精度の安定性が確認された.GPT-4o モデルは翻訳精度においても最も優れており, 特に自然言語を使用した場合には誤訳や意味のずれがほとんど発生しなかった. これにより, モデルの性能向上が翻訳の精度向上にも寄与していることが示された.

対話継続性の評価実験では,GPT-3.5-turbo では自然言語よりも YAML が 72% 低かったのに対し,GPT-4o-mini では 3%高く,GPT-4o では 7%高いという結果が得られた. したがって, 高精度なモデルにおいて YAML の優位性が確認された.YAML のプロンプトは長くなっても構造化されているため, 一貫した翻訳の流れを維持しやすく, 誤訳を抑える効果があった. 特に GPT-4o モデルでは,YAML を使用することで文脈を適切に保持しながら翻訳が行われ, 安定した結果が得られた. 一方で, 自然言語のプロンプトは長くなるほど文脈の理解が難しくなり, 翻訳の一貫性が低下する傾向が確認された.

6.4 まとめ

以上の結果から, 低精度なモデルを使用する場合には自然言語プロンプトの方が安定した結果を得やすいが, 高精度なモデルでは YAML プロンプトの方が一貫性のある対話や翻訳を可能にすることが示唆された. 今後の課題として,YAML プロンプトの設計を改善し, 低精度なモデルでも適切に処理できるよう最適化することで, 全体的な翻訳システムの精度向上が期待される.

第7章　おわりに

本研究では, 異なる LLM のモデルとプロンプト形式（YAML と自然言語）を用いた翻訳タスクにおいて, シナリオ逸脱頻度, 曖昧性検査精度, および翻訳精度を比較・分析した. その結果, 低精度なモデルでは自然言語プロンプトの方が解釈しやすく安定した翻訳が得られるのに対し, 高精度なモデルでは YAML プロンプトを用いることで一貫性のある翻訳が可能となることが示された. 特に, 同じ API に翻訳を続けて依頼するときは構造化された YAML を用いた場合に, より安定した翻訳結果が得られる傾向が確認された.

本研究の貢献は以下の通りである.

汎用的なシナリオ言語の定義

複数の大規模言語モデル (LLM) で利用可能な汎用的なシナリオ記述を実現するため, シンタックス (文法) とセマンティクス (意味) の明確な定義を行った. これにより, 特定のモデルに依存しない翻訳エージェントの設計が可能となり, システムの適用範囲を広げる基盤を提供した.

シナリオによる翻訳エージェントの制御

高精度なモデルでは, 同じ API に連続して翻訳を依頼した場合 YAML プロンプトの曖昧性の検査精度と翻訳精度の向上が確認できた. また, 構造化された YAML のプロンプトは, ユーザとの長距離インタラクションでもエージェントの性能が維持されることが示された.

今後の課題として, 低精度なモデルでも YAML を適切に理解できるようにプロンプトを改善することが挙げられる. また, どのモデルを使用しても, 翻訳エージェントが多義語や文化によって意味が変わる単語の曖昧性を正しく判断し, 翻訳の精度を向上させることが求められる. さらに, どのモデルを使用しても, 同じ API に連続して翻訳を依頼した場合でも,YAML を使用することで安定した翻訳ができるかどうかを検証することも重要な課題となる.

謝辞

本研究を行うにあたり，熱心なご指導，ご助言を賜りました村上陽平教授に深く感謝申し上げます．また，普段からお世話になっている社会知能研究室の皆様にも心より感謝申し上げます．

参考文献

[1] Chungi Shi, Toru Ishida, Donghui Lin: Translation Agent: A New Metaphor for Machine Translation, New Generation Computing, Vol.32, pp.163-186 (2014).

[2] Toru Ishida, Shohei Yamane: Introduction to Scenario Description Language Q, Second International Conference on Informatics Research for Development of Knowledge Society Infrastructure (ICKS'07), IEEE, pp.137-144(2007).

[3] 石田亨, 福本理人：インタラクション設計言語 Q の提案, 人工知能学会誌, 17 巻，2 号，SP-B，pp.166-168(2002).

[4] Wenxuan Zhou, Sheng Zhang, Hoifung Poon, Muhao Chen: Context-faithful Prompting for Large Language Models, arXiv, vol.2303, no.11315v2, pp.1-13 (2023).

[5] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, Chris Callison-Burch: Faithful Chain-of-Thought Reasoning, arXiv, vol.2301, no.13379v3, pp.1-25 (2023).

[6] Malin Eriksson, Victor Hallberg: Comparison between JSON and YAML for Data Serialization, Bachelor's Thesis in Computer Science, School of Computer Science and Engineering, Royal Institute of Technology, (2011).

[7] Dilshan I. De Silva, I. S. Gallage: The Role of Syntax and Semantics in Rule-Based Translation: A Comprehensive Review, International Conference on Information and Communication Technology for Development for Africa (ICT4DA),pp. 229–234 (2024).

付録

A.1　テスト文

5.2.3 データセットで説明したテスト文は, 以下の通りである.

- **語句の曖昧性あり**：100 文
  - **多義語**：60 文
    1. I went to the bank.
    2. I saw a bat.
    3. She gave him a ring.
    4. He picked up a bug.
    5. He took a bow.
    6. He played the field.
    7. She saw a match.
    8. He hit the post.
    9. I found a seal.
    10. He is on fire.
    11. I need a new handle.
    12. He saw the fall.
    13. He saw some bark.
    14. He saw a date.
    15. I saw a fly.
    16. I saw the note.
    17. I saw the GOAT climbing to the top.
    18. She saw the row.
    19. They saw the sink.
    20. I saw the spring.
    21. He saw the tie.
    22. I saw the trip.
    23. I saw one yard.
    24. I saw the wave.
    25. I saw the club.
    26. She saw my race.

27. I saw the draft.
28. I saw the band.
29. He saw the coach.
30. She saw the plot.
31. He saw the chest.
32. She saw the spike.
33. She set the net.
34. I saw the lap.
35. I saw the trolley.
36. I have plaster.
37. That ' s fly.
38. He raised the bar.
39. She took the lead.
40. He struck the nail.
41. I saw the scale.
42. She pointed at the light.
43. He measured the second.
44. I left the plant.
45. I studied the rock.
46. She hit the ground.
47. I cracked the nut.
48. I explored the mine.
49. I check the capital.
50. I saw the bill.
51. They saw the strike.
52. He saw the duck.
53. I saw a board.
54. She saw the crane.
55. He saw the horn.
56. I check the aspect.
57. She saw the letter.
58. She saw the case.

59. I saw the organ.
60. She saw the current.

– **文化依存の単語**：20 文

1. Your girlfriend is homely.
2. The baby was holding a dummy.
3. He bought some jelly at the store.
4. He decided to become a chemist after finishing school.
5. He started wearing braces last week.
6. She went to see the flat yesterday.
7. He put plaster on his arm.
8. She is wearing a jumper.
9. I have the casket.
10. I eat chips.
11. I wear pants.
12. I wear the vest.
13. I saw the trolley.
14. I go to a public school.
15. He plays football.
16. The old torch was left in the corner of the room.
17. She found a biscuit in the kitchen.
18. He wore braces.
19. We sat in the last carriage.
20. I saw boot.

– **口語表現の単語**：20 文

1. I failed my math test. OOF.
2. I forgot his homework again. SMH.
3. That joke was hilarious, LOL!
4. BRB, gotta grab some water.
5. TBH, I didn ’ t really like that movie.
6. I have to go now. TTYL!
7. That joke had me ROFL!
8. I have FOMO every time I miss a party.

9. Let's go skydiving! YOLO!
10. GG, you almost had me!
11. I didn ' t need to know that, TMI!
12. That concert was a W!
13. Spill the tea!
14. That ' s cap.
15. I ' m going to the party tonight. HBU?
16. NGL, I wasn ' t expecting that ending.
17. That meme was perfect, IYKYK.
18. FWIW, I think you should go for it.
19. I ' m so tired RN.
20. I ' ll be AFK for a few minutes.

- **語句の曖昧性あり**: 100 文
  1. It is cold today.
  2. She drinks water every morning.
  3. He eats breakfast at 7 AM.
  4. He plays the piano beautifully.
  5. The cake tastes sweet.
  6. The boy is reading a book.
  7. She writes in her notebook daily.
  8. The sun sets in the west.
  9. Birds fly in the sky.
  10. The flowers bloom in spring.
  11. They study at the library.
  12. He is fixing the broken chair.
  13. The baby sleeps peacefully.
  14. She is planting a tree in the garden.
  15. The clock on the wall is round.
  16. The river flows to the sea.
  17. She runs every morning in the park.
  18. The teacher explains the lesson clearly.
  19. The snow falls in winter.

20. The stars shine brightly at night.
21. The dog barks loudly.
22. She is sewing a new dress.
23. The students listen carefully to the lecture.
24. The computer is on the desk.
25. He is painting the fence white.
26. She enjoys swimming in the pool.
27. The grass is green in summer.
28. They are playing soccer on the field.
29. The fire burns brightly in the fireplace.
30. The chef cooks delicious meals.
31. She collects stamps as a hobby.
32. He drinks coffee with milk.
33. The wind blows gently in the evening.
34. The road is straight and wide.
35. The baby laughs when he sees his toy.
36. The fish swims in the clear water.
37. The farmer harvests rice in autumn.
38. The cat jumps onto the table.
39. He buys fresh vegetables from the market.
40. The girl is drawing a picture of a house.
41. She dances gracefully on the stage.
42. The horse gallops across the field.
43. The ice melts in the warm sun.
44. He rides his bicycle to work every day.
45. The lighthouse guides the ships at sea.
46. She is knitting a scarf for her friend.
47. The bridge connects the two towns.
48. The book contains many interesting stories.
49. He is folding the laundry neatly.
50. The rain falls gently on the roof.
51. The artist is creating a beautiful sculpture.

52. The leaves turn yellow in autumn.
53. She is baking bread in the oven.
54. The owl hoots in the forest at night.
55. The children are playing hide-and-seek in the yard.
56. He repairs watches in his small shop.
57. The airplane flies high above the clouds.
58. The baker sells freshly baked bread.
59. The bell rings loudly in the school.
60. The climbers reached the top of the mountain.
61. She arranges the flowers in a vase.
62. The farmer plows the field with his tractor.
63. He teaches math to high school students.
64. The turtle walks slowly on the sand.
65. The lamp on the table gives soft light.
66. She loves to watch the sunset by the lake.
67. The eagle soars high in the sky.
68. The shop sells handmade wooden toys.
69. The waiter serves the food with a smile.
70. The baker decorates the cake with fresh cream.
71. The singer performs on the stage every night.
72. The dog wags its tail when it is happy.
73. She is cleaning the windows with a cloth.
74. The cow grazes on the green grass.
75. He builds houses for a living.
76. The students are working on their homework.
77. The clock strikes twelve at midnight.
78. She organizes books on the shelf.
79. The squirrel hides nuts in the tree.
80. The candle glows softly in the dark.
81. The fisherman catches fish in the river.
82. She prepares breakfast every morning.
83. The musician tunes his guitar before the concert.

84. The train station is busy during the morning.
85. The children are learning to play the violin.
86. The old man tells stories to his grandchildren.
87. The rooster crows early in the morning.
88. The moon rises above the mountains.
89. She decorates the room with colorful balloons.
90. The butterfly lands on the flower.
91. The boy builds a sandcastle on the beach.
92. The cook washes the vegetables before cooking.
93. The bus stops at every corner in the city.
94. The chef prepares a special dish for the guests.
95. The water in the glass is cold and fresh.
96. The car stops at the red traffic light.
97. She arranges chairs in the hall before the event.
98. The teacher writes on the blackboard with chalk.
99. The gardener waters the plants in the morning.
100. The library is a quiet place for reading.