DialogFlow CX Intents

Allen Sanders

AGENDA

1. Intents in DialogFlow CX
2. Creating & Configuring Intents
3. Tips for Designing Effective Intents

01

Intents in DialogFlow CX

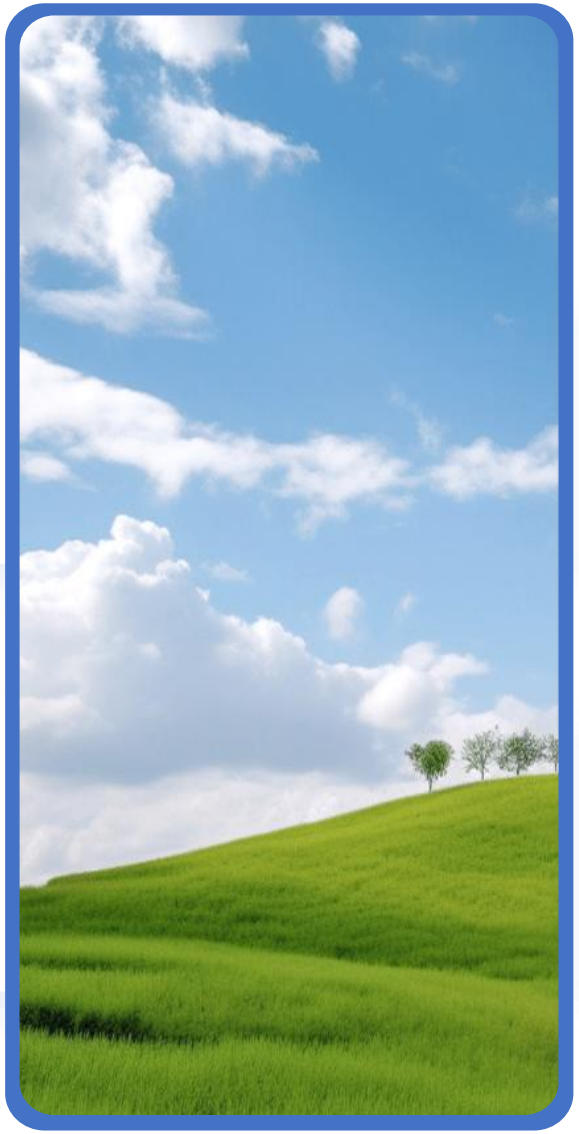
Intents - Definition

What is an Intent?

An intent represents a mapping between what a user says and the corresponding action in your application

Components of an Intent

Intents typically consist of training phrases, parameters, and responses that guide the interaction flow



Interaction & Dialogue

User Inputs

The user's expressions are matched to intents to decide the subsequent response and next steps in the dialogue

Agent Responses

Based on the matched intent, the agent responds appropriately, guiding the user through a structured conversation

Mapping User Inputs

Identifying User Intent

Techniques include collecting training phrases and using machine learning to predict user intents accurately

Handling Ambiguities

Strategies for managing scenarios where multiple intents may match; improving accuracy and user satisfaction

Comparison (ES vs. CX – Redux)

Differences in Architecture

DialogFlow CX uses a flow-based model which contrasts with the simpler intent-based model of DialogFlow ES

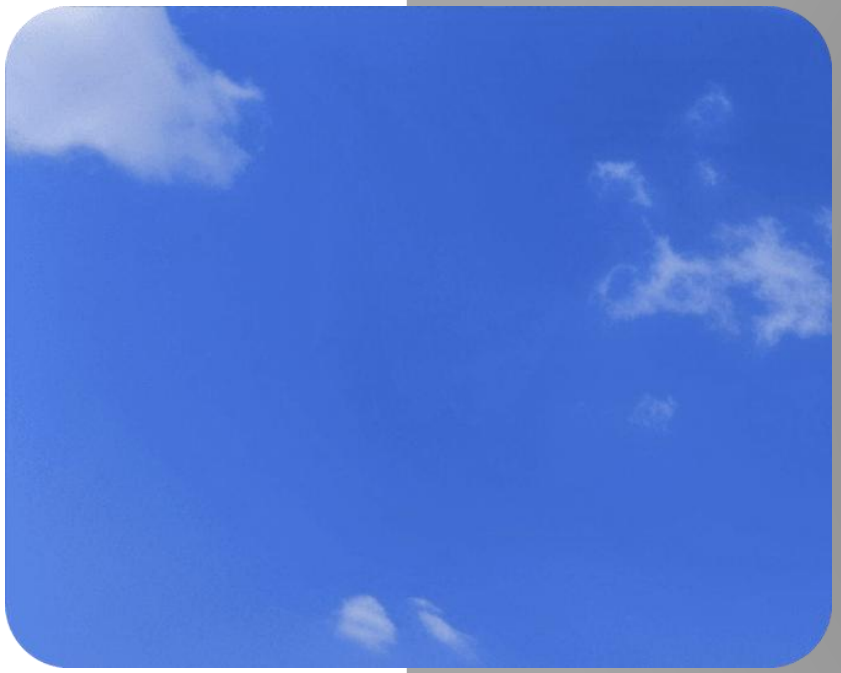
01

Scalability and Flexibility

Dialogflow CX supports larger, more complex projects compared to DialogFlow ES

02

Comparison (ES vs. CX – Redux)



Simple Applications

DialogFlow ES might still be the preferable choice for simplicity

Complex Interactions

DialogFlow CX excels due to its advanced tracking and management capabilities

Designing Effective Flows

Flow Strategy

Important to conceptualize conversation flows to ensure user intents are met effectively and intuitively

Flow Components

Distinguishing between various elements such as pages, parameters, and transition rules in designing flows is a critical activity

How Intents Help Enhance User Experience

Personalization

Using intents and user data to tailor responses to individual users, helps provide a customized and personalized conversation experience

Error Handling

There are techniques for managing errors or unexpected inputs effectively, which helps maintain a smooth interaction

Maintaining Intents

01 Continuous Improvement

Important to iteratively refine intents based on user feedback and interaction analytics

02 Version Control

Proper version management proves critical to keeping track of changes and ensuring stability in intent handling

Training Data/Phrases

Sources of Training Phrases

Gathering data from various user interactions can help to improve intent matching accuracy

Quality of Training Data

Ensuring that the collected phrases are diverse, and representative of real user queries gives the ML models more “fuel” for refinement

Utilizing Machine Learning

Algorithm Selection

Choosing appropriate machine learning algorithms to train your intent matching model pays dividends

Model Evaluation

Regularly evaluating your models to ensure they meet accuracy and performance standards is a valuable operational step

Action Mapping Techniques

Scripting Responses

It is important to craft responses that are clear, concise, and aligned with user expectations

Dynamic Response Generation

Generating responses dynamically based on context and user inputs can help you create responsive interactions

Best Practices – Clear & Distinct Intents

Avoiding Overlap

Proper strategies help to ensure that intents are clear and distinct to avoid confusion and improve matching accuracy

01

Simplifying Responses

Keeping responses simple and to the point can enhance user satisfaction and clarity

02

Managing Complexity

01

Using Sub-Flows

Breaking down complex interactions into sub-flows and smaller, modular “pieces” can help to improve manageability and scalability

02

Hierarchical Design

Organizing intents hierarchically to help streamline the design and make maintenance easier

Documentation & Updates

Keeping Documentation Current

Regularly update documentation to reflect changes and ensure team members are always informed of the latest implementation

Continuous Testing

Implement ongoing testing and validation to ensure that all intents function as expected

02

Creating & Configuring Intents

Intent Examples

Greeting Intent

Responding appropriately to user greetings like “Hello,” “Hi,” or “Good morning”

Complex Intent Example

Handling multi-step queries such as order tracking, like “Where is my order?” or “Check my delivery status”

Examples of Fallback Intents

Handling Unexpected Inputs

It is important to manage situations where the user input does not match any defined intents

Chained Intents Management

You may need to be able to transition smoothly between different intents within a single conversation

Accessing the Intents Panel

Console Navigation

Via lab, we will look at the step-by-step process of accessing and interacting with the intents panel in the DialogFlow CX console

UI Familiarity

Understanding the user interface elements of the intents panel helps you efficiently navigate during creation & configuration

Naming Best Practices

Clear Descriptive Names

- It is important to use names that accurately describe the intent, like "Order_Tracking_Intent“
- Identify a standard and practice it

Consistency in Naming

Maintaining a consistent naming scheme helps avoid confusion and improves ongoing management and maintenance

Defining Intent Scope

1

Global vs. Flow-Specific Intents

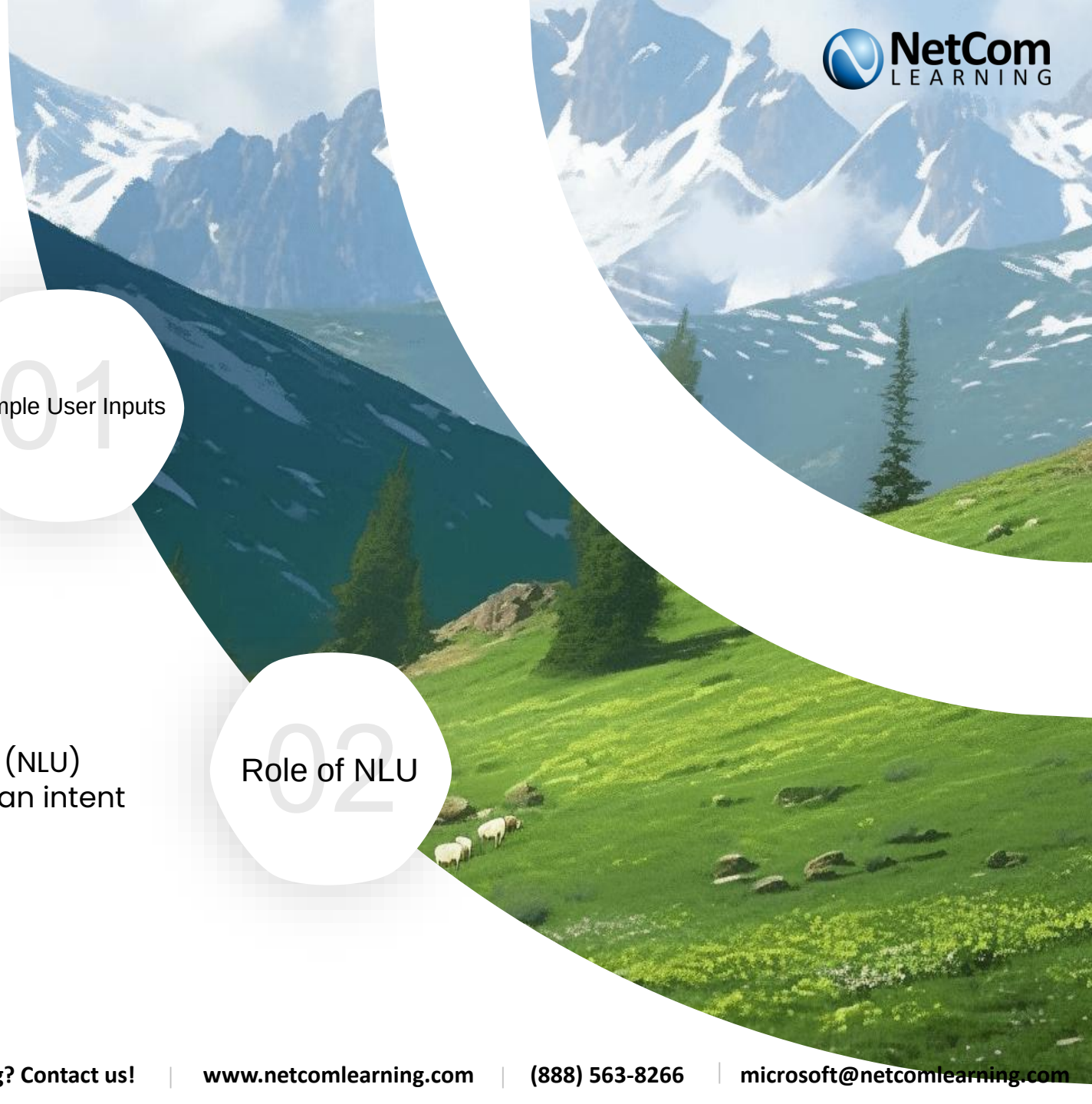
Understand the differences between global and flow-specific intents

2

Determining Scope

Decide on proper criteria for deciding whether an intent should be global or specific to a particular flow

What Are Training Phrases?



01 Sample User Inputs

Provides specific examples (and variations) of what users might say that should trigger an intent

02 Role of NLU

Natural Language Understanding (NLU) identifies and maps user input to an intent

Best Practices for Training Phrases

Natural Language Variations

Including a variety of natural language phrases helps account for different ways users might express the same intent

Avoiding Redundancy

Ensuring training phrases are diverse enough to avoid redundancy and excessive specificity provides a wider range of options for matching and fulfillment

Examples of Training Phrases

Booking Intent Phrases

Examples like “I want to book a flight,” “Can you reserve a ticket for me?”

Customizing Phrases

Customizing phrases according to the specific needs of the application will pay dividends in terms of matching and user experience

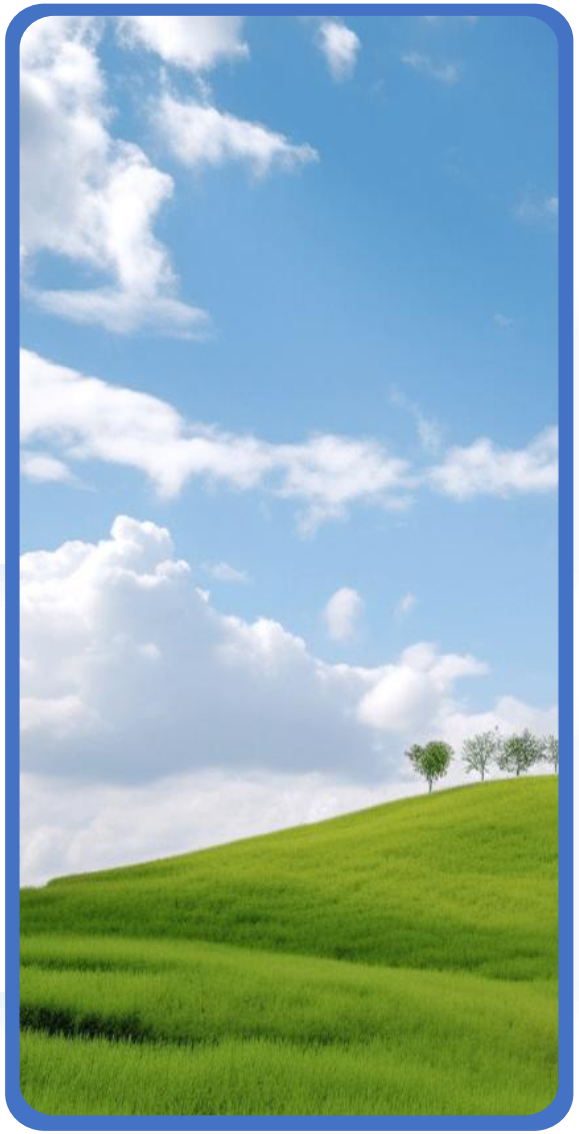
Response Types

Text-Based Responses

Use static and dynamic text responses for interacting with users

Rich Responses

Employ cards, images, and buttons to enhance user interaction



Using Parameters in Responses

Inserting dynamic data into responses, e.g., “Your order number {order_number} will arrive on {delivery_date}.”

Dynamic Data Insertion

Tailoring responses based on user history and context for a more personalized experience

Context-Aware Responses

Testing Responses

01.

Dialogflow CX Simulator

Utilize the Dialogflow CX simulator to test and refine intent responses

02.

Real-World Scenarios

Test intents in real-world scenarios to ensure they respond appropriately

Intent Routing Overview

Linking Intents

Connect intents to appropriate flows or pages within the DialogFlow CX environment to manage progression through a flow

Transitioning Intents

Defining clear transitions between intents helps maintain a coherent conversation flow

Setting Up Routing Conditions

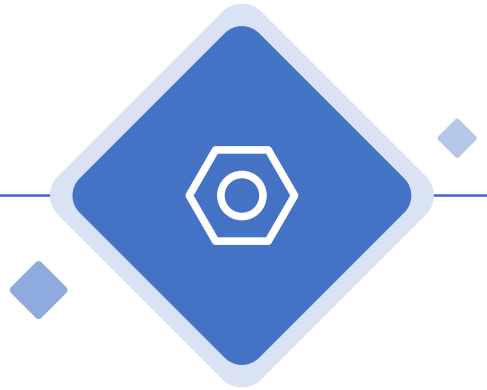
Conditional Routing

Define conditions for routing, e.g., if the user says, “cancel order,” route to the Order Cancellation flow

Handling Multiple Conditions

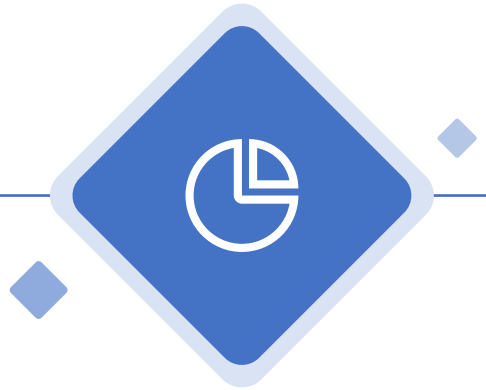
Verify that you are managing multiple routing conditions to ensure appropriate intent handling

Fulfillment Options



Cloud Functions / Webhooks

Use Cloud Functions or Webhooks to call external APIs for dynamic fulfillment



Database Lookups

You can fetch dynamic responses from a database to provide real-time information

Understanding Slot Filling

Collecting User Input

It is about gathering necessary user information during a conversation for mapping to structured inputs

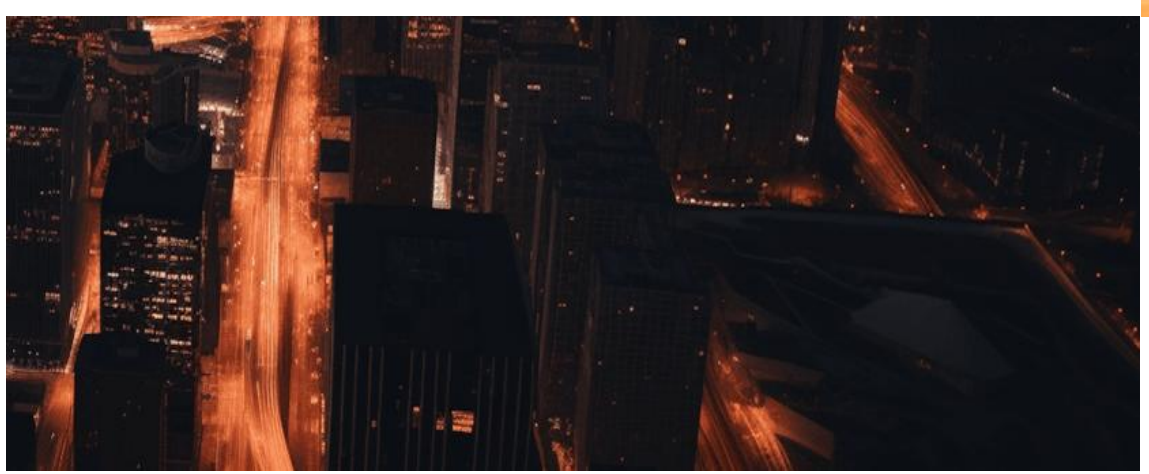
Managing Missing Entities

Handling situations where required entities are missing and prompting the user to fill slots ensures the necessary data is in place to complete a flow

Entity Extraction Techniques

Automatic Entity Detection

Utilize built-in NLU capabilities to automatically detect and extract entities



Custom Entities

Define and use custom entities that best match the needs of the particular conversation

Best Practices for Slot Filling

Clarifying User Inputs

Ask clarifying questions to ensure the accuracy of collected information

Maintaining Conversation Flow

Ensure the conversation remains smooth while collecting necessary data from the user

03

Designing Effective Intents

Diversify Training Phrases

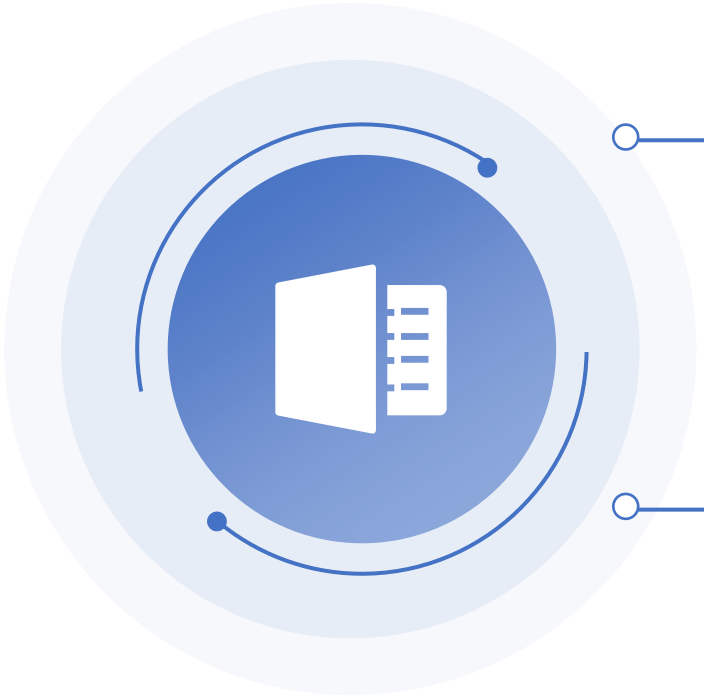
Capture Different Wording Styles

Include diverse examples to address various ways users phrase their queries, enabling broad understanding

Refining with User Data

Employ historical user interaction data to refine and enhance training phrases

Avoid Specific Keyword Reliance



Broad Phrasing Variety

Prevent over-dependence on particular keywords by incorporating a wide range of expressions

Scenario-based Testing

Test with diverse user inputs to ensure robustness across different scenarios

Utilize Testing & Test Cases

01

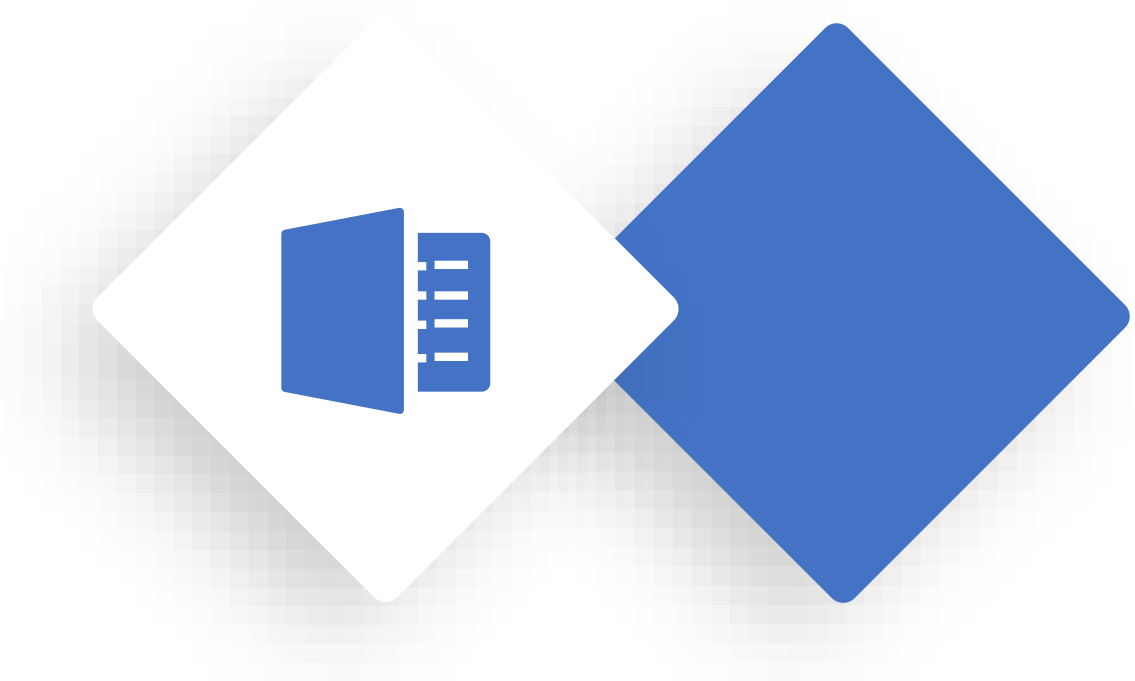
Missing Training Phrases Detection

Develop and use test cases to uncover gaps in your training data

02

Enhancement Cycle

Regularly update training phrases based on test case results



Implement Proper Fallback Intents

01

Handling Unknown Inputs

Create fallback intents to manage and direct unexpected user queries effectively

02

Dynamic Response Refinement

Use fallback recoveries to gather insights for refining primary intents

Analyze Misclassified Intents

Rectifying Training Data

Adjust training data and model parameters based on misclassification reports

Identifying Misclassifications

Utilize analytics to track and identify misclassified queries to improve intent accuracy

Ongoing Improvement

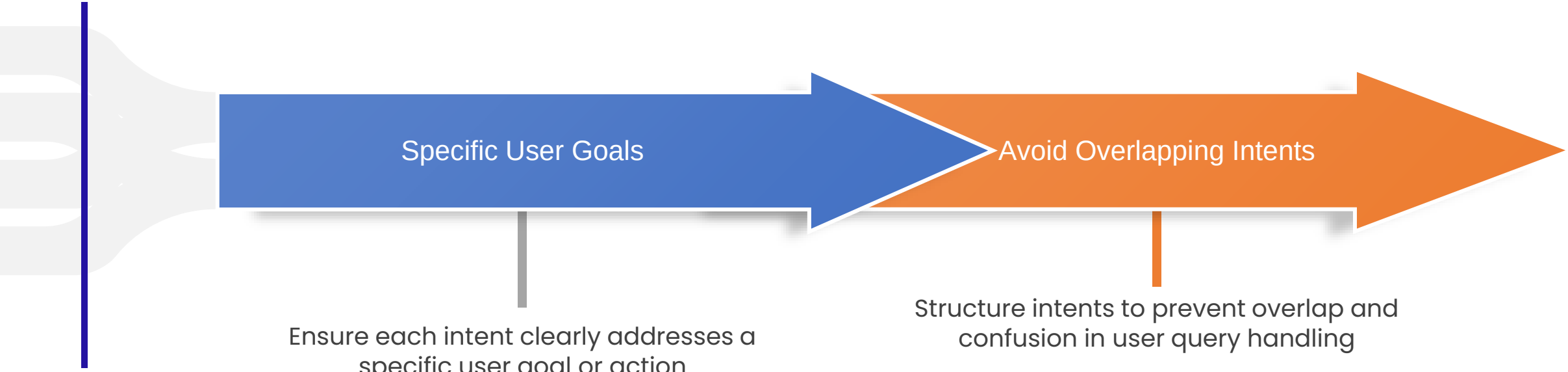
Real-World Data Utilization

Leverage real-world interaction data to continuously refine and improve intents

Continuous Monitoring

Set up monitoring tools to regularly evaluate intent performance

Clearly Defined Intents



Modular Approach

Reusable Components

Design intents with reusable components to streamline the structure and facilitate updates

Hierarchical Organization

Organize intents hierarchically to maintain clear relationships and dependencies

Context Utilization – Intent Accuracy

Contextual Precision

Employ contexts to manage conversation flow and maintain contextual accuracy.

Context Lifecycle Management

Monitor and manage context lifecycle to ensure relevant and timely responses.

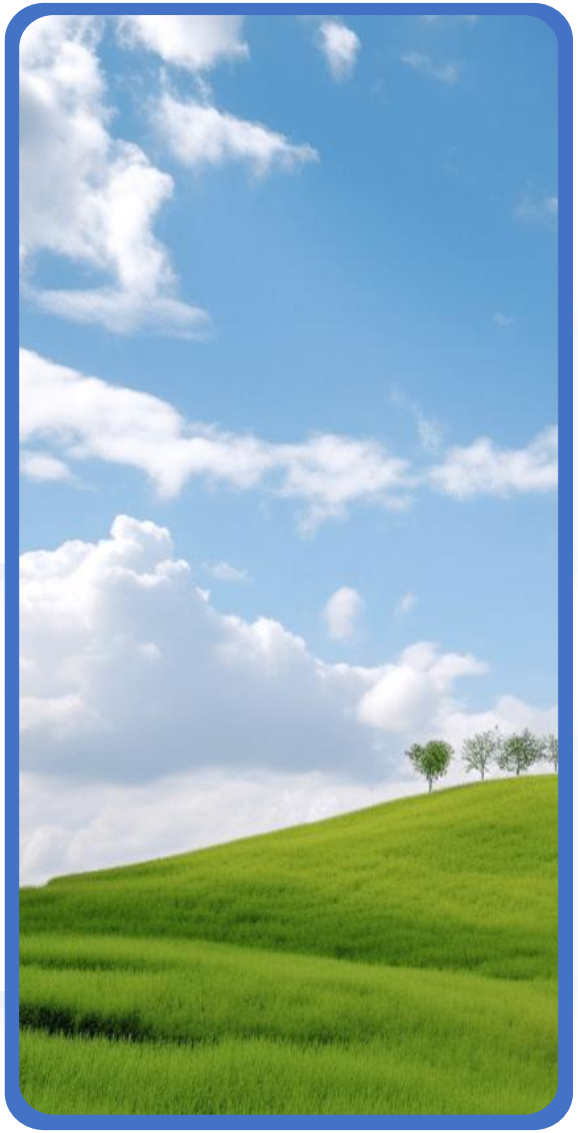
State Management

Contextual State Maintenance

Maintain conversational state to provide coherent responses based on prior interactions.

Contextual Layering

Use layered contexts to handle complex conversations with multiple interdependent steps.



Thank you