# AI Food Recommendation Chatbot: Multi-Agent RAG System

## System Overview

The AI Food Recommendation System is a sophisticated multi-agent conversational platform that combines natural language processing, machine learning, and advanced retrieval techniques to provide personalized food recommendations. The system employs a modular architecture with distributed components working in coordination to deliver contextually relevant suggestions through natural conversation.

## Architecture Components

### *Core System Architecture*

The system follows a **multi-agent orchestrator pattern** where specialized components handle distinct responsibilities while maintaining loose coupling. The architecture comprises four primary layers:

- **Conversation Layer**: Manages user interactions and dialogue flow
- **Recommendation Engine**: Processes queries and retrieves relevant suggestions
- **Data Layer**: Handles vector embeddings, user clustering, and data storage
  **Orchestration Layer**: Coordinates workflows across all system components

## Technology Stack

### *Large Language Model Integration:*

- OpenAI GPT-4o-mini for intent classification, slot extraction, and contextual reranking
- Temperature-controlled responses (0.1-0.4) for consistency
- Rate limiting implementation for API call management

### *Vector Database Infrastructure:*

- ChromaDB for distributed vector storage and similarity search
- Sharded architecture supporting horizontal scaling across 8 shards
- HuggingFace sentence-transformers/all-MiniLM-L6-v2 for text embeddings
- Support for 1M+ document capacity with 150k documents per shard

### *Machine Learning Pipeline:*

- K-means++ clustering algorithm for user segmentation
- Scikit-learn pipeline with 10-dimensional feature space
- User profiling based on demographics, preferences, and behavioral patterns
- Silhouette score optimization for cluster validation

## Detailed Component Analysis

### *Conversation Layer:*

**OpenAIConversationAgent** (conversation_agent.py):
The primary orchestrator managing dialogue flow through systematic state progression. Implements continuous conversation flow without terminating sessions, maintaining context across multiple turns.

Key features:

- Dual slot update mechanisms: new_query for complete context replacement, slot_updation for incremental updates
- Integration with query enhancement and vector retrieval pipelines
- Comprehensive error handling with fallback responses
- Question history tracking and conversation state management

**OpenAIIntentClassifier** (intent_classifier.py):
Utilizes OpenAI GPT-4o-mini to classify user intents across seven categories:

- RECOMMEND, FILTER_UPDATE, CLARIFICATION, FEEDBACK, GREETING, GOODBYE, OTHER
- Fallback pattern matching for API failures
- Confidence scoring with threshold-based decision making

**ConversationMemory** (memory.py):
Stateful memory management system tracking dialogue progression:

- Slot-based information storage with 7 predefined slots (dietary, cuisine_1, cuisine_2, item_name, price, meal_type, label)
- Context preservation mechanisms for maintaining conversation coherence
- Turn-by-turn history with timestamp tracking and slot update logging

**ResponseGenerator** (response_generator.py):
Systematic response generation following a clear progression pattern:

- State-driven dialogue flow: item preferences → dietary requirements → budget → search confirmation
- JSON-structured response parsing with fallback mechanisms
- Question deduplication to prevent repetitive interactions

**Slot Extraction System** (slot_extract.py):
Advanced entity extraction combining OpenAI LLM capabilities with rule-based fallbacks:

- Intent-aware slot filling with context preservation
- Support for cuisine mapping across 70+ cuisine types
- Flavor enhancement and dietary restriction handling
- Null value management preventing generic responses

### *Recommendation Engine*

**OpenAIQueryEnhancer** (query_enhancer.py):
LLM-powered query refinement system that transforms user preferences into optimized search queries:

- Semantic query construction combining cuisine types, specific dishes, and preferences
- ChromaDB-compatible filter generation with complex boolean logic
- Price handling with strict, approximate, and range conditions
- Cuisine mapping across comprehensive taxonomy of food types

**ShardedRetrievalAgent** (shards_retrieval.py):
Distributed vector search system managing queries across multiple ChromaDB shards:

- Parallel shard querying with configurable top-k results per shard
- Automated result aggregation and deduplication
- Support for complex metadata filtering

**Two-Stage Contextual Reranker** (rerank.py):
Advanced reranking system using dual-stage LLM evaluation:

*Stage 1: Context Analysis*

- Temporal context analysis examining eating progression patterns
- Document inventory mapping with cuisine categorization
- Adaptive ranking condition generation based on user journey
- Comprehensive document evaluation with mandatory reasoning

*Stage 2: Final Ranking*

- Top-10 document selection based on contextual scoring
- Multi-criteria evaluation (critical, high, medium, low priority)
- Quality assurance validation for ranking coherence
- Fallback ranking using rating-price ratio for API failures

## *Data Layer*

**Embeddings Setup** (embeddings.py):
Vector embedding infrastructure optimized for CPU deployment:

- sentence-transformers/all-MiniLM-L6-v2 model with normalized embeddings
- Memory-aware configuration based on available system resources
- CPU optimization for cost-effective cloud deployment

**User Clustering Agent** (user_clustering_agent.py):
Machine learning model for user segmentation and personalization:

- K-means++ clustering with 4 user personas
- 10-dimensional feature space: Age, Monthly_Income, Gender, Weather, Marital_Status, Average_Rating, Average_Z_Score, purchase_sensitivity, C_Type, Veg_Ratio
- Silhouette score of 0.279 indicating reasonable cluster separation
- Default value handling for missing user attributes

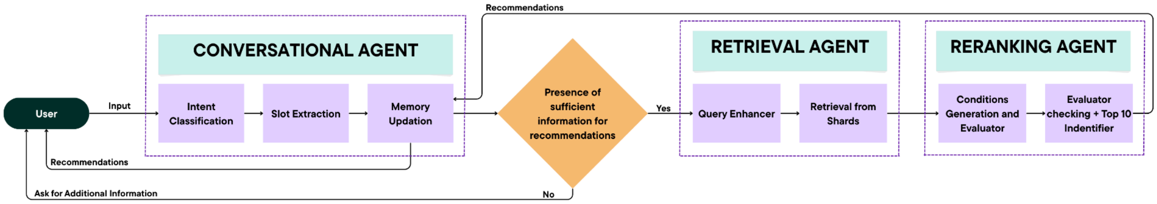
## *Orchestration Layer*

**RecommenderOrchestrator** (orchestrator.py)
Central coordination hub managing end-to-end recommendation workflow:

- Multi-agent coordination with shared memory state
- Configuration-driven component initialization
- Recommendation context handling with history tracking

# Technical Implementation Details

## *Data Flow Architecture*



The system processes user requests through a sophisticated pipeline:

1. **Input Processing**: User utterances are captured and normalized
2. **Intent Classification**: OpenAI determines user intent with confidence scoring
3. **Slot Extraction**: Entities are extracted and stored in conversation memory
4. **Query Enhancement**: LLM refines search queries with semantic optimization
5. **Vector Retrieval**: Distributed search across sharded ChromaDB instances
6. **Contextual Reranking**: Two-stage evaluation for relevance optimization
7. **Response Generation**: Systematic dialogue progression with context awareness

## *Vector Database Sharding Strategy*

The system implements horizontal partitioning across 8 ChromaDB shards:

- **Shard Distribution**: 150,000 documents per shard.
- **Collection Strategy**: Individual collections per shard with persistent storage
- **Query Parallelization**: Concurrent querying across all shards with result aggregation
- **Fault Tolerance**: Individual shard failure handling without system-wide impact

## *Memory Management and State Persistence*

**Conversation State Tracking**:

- Slot-based information storage with null value initialization
- Context-aware updates preserving existing information
- Turn-by-turn history with comprehensive metadata
- Session persistence across conversation boundaries

**User Profile Integration**:

- Demographic and behavioral feature extraction
- Real-time cluster assignment for new users

- Preference learning from conversation context
- Historical interaction tracking

## *LLM Integration Patterns*

**Multi-Temperature Strategy**:

- Intent classification: 0.1 for consistency
- Slot extraction: 0.2 for balanced creativity
- Response generation: 0.4 for natural language variety
- Reranking: 0.1 for deterministic evaluation

**Rate Limiting and Error Handling**:

- 60 requests per minute throttling
- Exponential backoff for API failures
- Fallback mechanisms for each LLM-dependent component
- Graceful degradation maintaining system availability

## *Reranking Algorithm Deep Dive*

The two-stage reranking system implements sophisticated contextual evaluation:

**Stage 1 Analysis Framework**:

- **Temporal Context**: Analyzes time gaps and eating progression patterns
- **Document Inventory**: Maps cuisine types, price ranges, and quality tiers
- **Condition Generation**: Creates measurable ranking criteria based on user journey
- **Comprehensive Evaluation**: Provides reasoning for every retrieved document

**Stage 2 Ranking Logic**:

- **Critical Conditions**: Exclusionary criteria for top-3 rankings
- **High Priority**: Creates clear ranking tiers
- **Tie-Breaking**: Uses medium/low conditions for fine-tuning
- **Quality Assurance**: Validates ranking logic and journey alignment

# Data Processing and Feature Engineering

## *User Clustering Model*

The system employs a sophisticated user segmentation approach:

**Feature Engineering**:

- Age and income standardization
- Behavioral metrics (rating patterns, purchase sensitivity)
- Demographic encoding (gender, marital status, weather preference)
- Cuisine type preferences and vegetarian ratios

**Clustering Pipeline**:

- K-means++ initialization for stable clustering
- 4-cluster solution with balanced distribution
- Silhouette analysis for cluster validation
- Persona mapping for interpretability

**User Personas Identified**:

1. Established Urban Professionals
2. Premium Self-Employed Segment
3. Young Urban Students
4. Price-Sensitive Employees

## *Vector Embedding Strategy*

**Document Processing**:

- Text normalization and preprocessing
- Metadata extraction and enrichment
- Semantic embedding generation using sentence transformers
- Vector storage with comprehensive metadata

**Query Processing**:

- Real-time embedding generation for user queries
- Semantic similarity computation using cosine distance
- Multi-field search across content and metadata
- Result scoring and relevance ranking

# System Integration and Workflow

## *End-to-End Recommendation Flow*

1. **User Profile Initialization**:
    - Demographic data collection
    - Cluster assignment using trained ML model
    - Preference initialization in conversation memory
2. **Conversation Management**:
    - Natural language intent detection
    - Progressive slot filling with context preservation
    - Systematic dialogue state progression
    - Multi-turn conversation handling
3. **Query Processing**:
    - Semantic query construction from conversation context
    - LLM-enhanced query refinement
    - Complex filter generation for vector search
4. **Retrieval and Ranking**:
    - Distributed vector search across sharded database
    - Contextual reranking with temporal analysis
    - Top-10 recommendation selection
5. **Response Delivery**:
    - Recommendation presentation with reasoning
    - Follow-up question generation
    - Conversation continuity management

# Performance Characteristics

## *Retrieval Performance*

- **Vector Search**: Sub-second response times across 1M+ documents
- **Sharded Architecture**: Linear scalability with shard addition
- **Caching Strategy**: Embedding caching for frequently accessed queries

### *LLM Integration Efficiency*

- **Request Optimization**: Batch processing where possible
- **Context Management**: Efficient prompt construction and token usage
- **Fallback Mechanisms**: Reduced API dependency through local fallbacks

### *Memory and Resource Usage*

- **CPU-Optimized**: Designed for cost-effective deployment
- **Memory Management**: Efficient embedding storage and retrieval
- **Scalable Architecture**: Horizontal scaling capabilities

## Quality Assurance and Validation

### *Recommendation Quality*

- **Two-Stage Validation**: Contextual analysis and ranking verification
- **User Journey Alignment**: Recommendations match conversation flow
- **Diversity Metrics**: Balanced recommendation sets across categories

### *Conversation Quality*

- **Intent Classification Accuracy**: Multi-tier validation with fallbacks
- **Slot Filling Precision**: Entity extraction with confidence scoring
- **Response Relevance**: Context-aware response generation

### *System Reliability*

- **Error Handling**: Comprehensive exception management
- **Graceful Degradation**: Partial functionality during component failures
- **Recovery Mechanisms**: Automatic retry and fallback systems

This technical architecture represents a comprehensive approach to building an AI-powered food recommendation system that balances sophistication with practical deployment considerations. The modular design, advanced ML techniques, and robust error handling create a production-ready system capable of delivering personalized food recommendations through natural conversation.