**Arquitectura Híbrida: Sistema Experto + Red Neuronal**

Propuesta de Integración para Optimización de Recomendaciones

# **1. Arquitectura General**

El sistema propuesto combina las fortalezas de un Sistema Experto (basado en reglas CLIPS) con una Red Neuronal (para aprendizaje y optimización), creando una arquitectura híbrida que ofrece tanto explicabilidad como capacidad de aprendizaje.

Estructura del Sistema:

┌─────────────────────────────────────────────────────────────┐  
│ SISTEMA HÍBRIDO │  
│ │  
│ ┌──────────────────┐ ┌──────────────────────┐ │  
│ │ Sistema Experto │ │ Red Neuronal │ │  
│ │ (CLIPS) │◄────────┤ (Optimizador) │ │  
│ │ │ │ │ │  
│ │ • Reglas │ │ • Aprende pesos │ │  
│ │ • Inferencia │ │ • Feedback loop │ │  
│ │ • Puntuación │ │ • Ajuste continuo │ │  
│ └──────────────────┘ └──────────────────────┘ │  
│ │ │ │  
│ └──────────┬─────────────────┘ │  
│ ▼ │  
│ ┌──────────────────┐ │  
│ │ Recomendación │ │  
│ │ Final │ │  
│ └──────────────────┘ │  
└─────────────────────────────────────────────────────────────┘

# **2. Cómo Funcionaría**

## **2.1. División de Responsabilidades**

* Sistema Experto (CLIPS) - Motor de Inferencia:
* • Aplica reglas (filtros, puntuación, penalizaciones)
* • Genera justificaciones explicables
* • Ordena restaurantes por índice U
* • Mantiene la transparencia del sistema
* Red Neuronal - Optimizador de Pesos:
* • Aprende los pesos óptimos (wg, wp, wd, wq, wa)
* • Ajusta según feedback del usuario
* • Se adapta con el tiempo
* • Mejora la personalización

## **2.2. Flujo de Funcionamiento**

Fase 1: Recomendación Inicial

Usuario hace búsqueda  
 ↓  
Red Neuronal predice pesos óptimos basado en:  
 • Historial de feedbacks previos  
 • Características del usuario (presupuesto, gustos, etc.)  
 • Contexto (clima, franja horaria)  
 ↓  
Sistema Experto recibe:  
 • Usuario + Pesos optimizados por NN  
 • Restaurantes  
 • Contexto  
 ↓  
CLIPS aplica reglas y genera recomendaciones  
 ↓  
Se muestran resultados al usuario

Fase 2: Aprendizaje (Feedback Loop)

Usuario selecciona un restaurante (o rechaza)  
 ↓  
Backend recibe feedback:  
 • Restaurante seleccionado  
 • Restaurantes rechazados (alternativas no elegidas)  
 ↓  
Red Neuronal:  
 1. Extrae características de seleccionado vs rechazados  
 2. Calcula pesos "ideales" para este usuario  
 3. Entrena con backpropagation  
 4. Actualiza modelo  
 ↓  
Modelo mejorado para próximas búsquedas

# **3. Diseño de la Red Neuronal**

## **3.1. Arquitectura**

Estructura propuesta:

* Capa de Entrada (5 características):
* • Afinidad (coincidencia de cocinas): normalizado 0-1
* • Precio relativo: normalizado 0-1
* • Cercanía: normalizado 0-1
* • Calidad (rating): normalizado 0-1
* • Disponibilidad: normalizado 0-1
* Capa Oculta: 8 neuronas con activación ReLU
* Capa de Salida: 5 neuronas con activación Softmax (pesos normalizados)
* • wg (gustos)
* • wp (precio)
* • wd (distancia)
* • wq (calidad)
* • wa (disponibilidad)

## **3.2. Entrenamiento**

* Tipo: Supervisado
* Entrada: Características extraídas (usuario/restaurante/contexto)
* Salida esperada: Pesos ideales calculados del feedback
* Algoritmo: Backpropagation + Descenso de Gradiente
* Learning Rate: 0.01 (conservador)

# **4. Ventajas de la Arquitectura Híbrida**

1. 1. Explicabilidad: CLIPS sigue generando justificaciones claras y comprensibles
2. 2. Aprendizaje: La red aprende preferencias sin reglas fijas
3. 3. Personalización: Ajusta pesos según el perfil del usuario
4. 4. Adaptabilidad: Mejora con el tiempo
5. 5. Robustez: Si la red falla, CLIPS funciona con pesos por defecto

# **5. Plan de Implementación**

## **5.1. Backend (Python/FastAPI)**

1. 1. Módulo neural\_network.py:

* • Clase WeightOptimizerNN
* • Forward pass (predicción)
* • Backpropagation (entrenamiento)
* • Guardado/carga de modelo

1. 2. Integración en main.py:

* • Endpoint /api/recommend: Usa NN para predecir pesos antes de CLIPS
* • Endpoint /api/feedback: Recibe selecciones, entrena la red
* • Persistencia del modelo en nn\_model.json

1. 3. Historial de feedbacks:

* • user\_feedback\_history.json para análisis y debugging

## **5.2. Frontend (Opcional)**

* • Botón "Esta recomendación fue útil" para feedback explícito
* • Tracking implícito: cuando el usuario hace clic en un restaurante

# **6. Ejemplos de Funcionamiento**

Escenario 1: Usuario nuevo (sin historial)

NN → Pesos por defecto: [0.35, 0.20, 0.25, 0.15, 0.05]  
CLIPS → Recomendación estándar  
Usuario → Selecciona restaurante con alta afinidad, precio medio  
NN aprende → Aumenta wg, disminuye wp

Escenario 2: Usuario recurrente

NN → Ya aprendió que prefiere calidad sobre precio  
Pesos: [0.30, 0.10, 0.20, 0.35, 0.05] (mayor wq)  
CLIPS → Recomendaciones más orientadas a rating

# **7. Consideraciones Importantes**

1. 1. Datos iniciales: Puede comenzar con pesos por defecto y aprender gradualmente
2. 2. Cold start: Usuarios nuevos usan pesos por defecto
3. 3. Overfitting: Limitar complejidad de la red (8 neuronas es razonable)
4. 4. Interpretabilidad: Los pesos finales se pueden mostrar al usuario

# **8. Alternativas Más Simples (si no quieres NN completa)**

1. 1. Ajuste Heurístico de Pesos:

* • Aumentar wg si selecciona restaurantes con alta afinidad
* • Aumentar wp si rechaza restaurantes caros
* • No requiere numpy ni backpropagation

1. 2. Sistema de Reglas Adaptativas:

* • Reglas en CLIPS que ajusten pesos según historial
* • Más simple, mantiene todo en CLIPS

1. 3. Ensemble:

* • Varios modelos (CLIPS con diferentes pesos) y votación
* • Usuario selecciona el mejor conjunto

# **9. Conclusión**

Esta arquitectura híbrida combina lo mejor de ambos mundos: la explicabilidad y transparencia del Sistema Experto con la capacidad de aprendizaje y adaptación de una Red Neuronal. El sistema puede comenzar funcionando con reglas y pesos por defecto, y gradualmente mejorar sus recomendaciones aprendiendo de las interacciones de los usuarios.

Para el trabajo académico, esto demuestra:

* • Diseño conceptual de red neuronal
* • Integración con Sistema Experto
* • Prototipo funcional básico
* • Fortalecimiento del núcleo del Sistema Experto