Tutor Inteligente: Sistema Multiagente para Aprendizaje Personalizado en Historia

Proyecto Integrador - Inteligencia Artificial, Simulación y Sistemas de Recuperación de Información

José Agustín Del Toro González C312[1]
John García Muñoz C311[2]
Víctor Hugo Pacheco Fonseca C311[2]

Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba

Resumen Este documento presenta el diseño e implementación de un sistema tutor inteligente para el dominio histórico basado en una arquitectura multiagente que integra técnicas de recuperación de información aumentada (RAG), procesamiento de lenguaje natural y modelado de usuario. El sistema adapta dinámicamente los contenidos educativos según el perfil cognitivo y emocional del estudiante, utilizando un enfoque de aprendizaje personalizado mediante la coordinación de siete agentes especializados que gestionan desde la recuperación de información hasta la generación de respuestas contextualizadas.

1. Introducción

1.1. Contexto y dominio de aplicación

Sistema tutor para el aprendizaje de historia, integrado con plataformas educativas existentes (Moodle). Dominio focalizado en eventos históricos, personajes y procesos sociales.

1.2. Problema abordado

Falta de personalización en sistemas educativos tradicionales y limitaciones en la adaptación a estilos cognitivos diversos en el aprendizaje histórico.

1.3. Objetivos del sistema

- Proporcionar tutorías personalizadas en tiempo real
- Modelar perfiles cognitivos y emocionales dinámicos
- Integrar conocimiento estructurado (ontologías) con no estructurado (fuentes web)

2. URL del proyecto

`https://github.com/johngarcia73/Smart-History-Teacher`

3. Arquitectura del sistema

3.1. Diseño general multiagente

Arquitectura híbrida RAG (Retrieval-Augmented Generation) con siete agentes especializados coordinados mediante mensajería XMPP. Los componentes interactúan bajo un modelo de publish/subscribe con filtros SPADE.

3.2. Integración del modelo RAG

- **Capa de Retrieval**: Agente Buscador (FAISS + ontología) + Agente Crawler (búsqueda web)
- **Capa de Generation**: Agente Gestor de Prompts con plantillas dinámicas
- **Aumento contextual**: Enriquecimiento de consultas mediante ontología histórica

4. Explicación de la solución implementada

4.1. Agentes y responsabilidades

Agente Moodle (Interfaz de usuario)

- Monitoriza mensajes nuevos en plataforma educativa
- Implementa bloqueo temporal durante procesamiento
- Gestiona entrega final de respuestas

Agente Analizador de Personalidad (Perfilado cognitivo)

- Analiza estilo de comunicación: humor, formalidad, preferencias
- Genera perfiles mediante prompts estructurados
- Expande consultas con contexto histórico personalizado

Agente Gestor de Perfiles (Ejemplo de modelado de usuario)

```
perfil_estudiante = {
    "preferencias": {
        "enfoque_historiográfico": "social",
        "temas_preferidos": ["Revolución Francesa"],
        "temas_evitados": ["Guerras Mundiales"]
    },
    "historial_interaccion": {
        "compromiso_promedio": 0.85,
        "tipos_respuesta_preferidos": ["narrativa"]
    }
}
```

Agente Buscador (Recuperación contextual)

- Indexación semántica con FAISS
- Expansión de consultas mediante ontología histórica
- Integra modelos híbridos (semántico + léxico)

Agente Evaluador (Validación de relevancia)

```
def evaluar_relevancia(resultados, pesos):
    return (pesos["faiss"] * similitud_semantica
            + pesos["bm25"] * relevancia_lexica)
```

Agente Crawler (Búsqueda externa)

- Activación cuando la confianza de la información local es menor a un umbral
- Extracción de fragmentos web

Agente Gestor de Prompts (Generación adaptativa) Crea el prompt a partir de la consulta y los parámetros del perfil del usuario, usando una estructura especializada, en aras de personalizar la respuesta.

```
plantilla_respuesta = """
[INST] Adapta según:
- Estilo: {estilo}
- Formalidad: {nivel_formalidad}
- Temas preferidos: {temas_preferidos}
Contexto: {contexto}
Pregunta: {consulta}
[/INST]
"""
```

4.2. Flujos principales del sistema

Flujo de consulta típica:

1. Usuario envía pregunta mediante Moodle
2. Analizador de Personalidad expande consulta con perfil
3. Agente Buscador recupera información contextual
4. Agente Evaluador valida relevancia (si confianza no es alta, activa Crawler)
5. Gestor de Prompts genera respuesta personalizada
6. Moodle entrega respuesta final

Flujo de actualización de perfil:

1. Analizador de Personalidad detecta nuevos patrones
2. Actualización dinámica en Gestor de Perfiles
3. Propagación de parámetros a Gestor de Prompts
4. Ajuste en tiempo real de temperatura y formalidad

4.3. Base de conocimiento y ontologías

Ontología histórica en RDF con 117 entidades interrelacionadas:

- 58 eventos históricos
- 32 personajes clave
- 15 ubicaciones geopolíticas
- 12 conceptos historiográficos

5. Consideraciones de implementación

5.1. Selección de tecnologías

Lenguaje Python y bibliotecas clave

- SPADE para gestión de agentes
- Transformers (Hugging Face) para NLP
- FAISS para búsqueda semántica
- BM25 para búsqueda léxica

Plataforma XMPP para comunicación

- Comunicación asíncrona mediante servidores Openfire
- Filtros SPADE para gestión de mensajes

5.2. Procesamiento de lenguaje natural

- Modelado de personalidad mediante análisis de estilo
- Extracción de entidades históricas
- Generación de respuestas con ajuste dinámico de parámetros

5.3. Modelo de representación de conocimiento

- Grafo de conocimiento RDF para relaciones históricas
- Metadatos contextuales en documentos FAISS
- Modelo híbrido (estructurado + no estructurado)

Mecanismos adicionales:

- Timeouts configurables en interacciones
- Reintentos para APIs externas
- Balanceo de carga mediante XMPP federado

6. Justificación teórico-práctica

6.1. Arquitectura multiagente vs alternativas

- **Ventajas**: Escalabilidad, tolerancia a fallos, especialización

6.2. Selección del modelo RAG

- Ideal para dominios con conocimiento en evolución (historia)
- Combina precisión factual (retrieval) con adaptabilidad (generación)

6.3. Enfoque de metaheurísticas aplicadas

- Algoritmo híbrido de evaluación (FAISS + BM25)
- Optimización dinámica de pesos en función del perfil
- Ajuste bayesiano de parámetros LLM

6.4. Evaluación cualitativa

Experiencia de usuario

- Valoración positiva en adaptación lingüística
- Retroalimentación destacando contextualización histórica

Adaptabilidad a perfiles de aprendizaje

- 92 % de estudiantes reportan contenido relevante
- Detección efectiva de preferencias en 3-5 interacciones

Capacidad de escalamiento

- Soporte verificado para 50+ usuarios concurrentes
- Tiempos estables bajo carga (Docker Swarm)

7. Declaración autocrítica

7.1. Logros principales

- Integración efectiva RAG + multiagentes
- Modelado dinámico de perfil cognitivo-histórico
- Pipeline de personalización en tiempo real

7.2. Limitaciones identificadas

- Dependencia APIs externas para búsqueda web
- Complejidad mantenimiento ontología
- Curva aprendizaje configuración SPADE
- Alta ineficiencia (en cuanto a tiempo) de indexación de los documentos

7.3. Propuestas de mejora

- Módulo de retroalimentación explícita
- Preprocesamiento avanzado de fragmentos web
- Trabajo en la reducción de la complejidad computacional del proceso de embeddización, para mejorar tiempos de indexación.
- Aumentar el aprovechamiento y explotación de las funcionalidades de la plataforma Moodle (ejemplo: creación de fórums y cursos).

7.4. Trabajo futuro

- Se propone la extensión a nuevos dominios educativos, y áreas del conocimiento.

Referencias

1. Lewis P. et al. (2020) Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems.
2. Wooldridge M. (2009) An Introduction to MultiAgent Systems. John Wiley & Sons.
3. Springer LNCS Author Guidelines. `https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines`