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 C-312
John García Muñoz C-311
Víctor Hugo Pacheco Fonseca C-311

Facultad de Matemática y Computación, Universidad de La Habana

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 por Generación (RAG), procesamiento de lenguaje natural (PLN) y modelado de usuario. El sistema adapta dinámicamente los contenidos educativos según el perfil cognitivo y emocional del estudiante, mediante la coordinación de siete agentes especializados que gestionan desde la recuperación de información hasta la generación de respuestas contextualizadas. La solución implementada demuestra alta capacidad de personalización, escalabilidad y eficacia en la entrega de tutorías históricas, validada mediante evaluaciones cuantitativas y cualitativas. El código fuente se encuentra disponible en un repositorio público de GitHub.

1. Introducción

1.1. Contexto y dominio de aplicación

El proyecto se enmarca en el desarrollo de sistemas tutores inteligentes (ITS) para la enseñanza de la historia. Se integra con plataformas educativas existentes, específicamente Moodle, y se enfoca en eventos históricos, personajes relevantes y procesos sociales. El dominio de aplicación exige manejo de información factual, contextualización histórica y adaptación a diversos estilos de aprendizaje.

1.2. Problema abordado

Se identificaron dos problemas principales: (1) la falta de personalización en sistemas educativos tradicionales, que ofrecen respuestas genéricas sin considerar el perfil del estudiante, y (2) las limitaciones en la adaptación a estilos cognitivos diversos, particularmente en el aprendizaje de historia donde se requiere contextualización y manejo de múltiples perspectivas.

1.3. Objetivos del sistema

Los objetivos planteados fueron:

- Proporcionar tutorías personalizadas en tiempo real mediante diálogos interactivos
- Modelar perfiles cognitivos y emocionales dinámicos que capturen preferencias y estilos de aprendizaje
- Integrar conocimiento estructurado (ontologías) con no estructurado (fuentes web)
- Diseñar una arquitectura escalable basada en agentes para garantizar modularidad y extensibilidad

2. URL del proyecto

El código fuente, documentación y recursos del proyecto están disponibles en el repositorio público:
`https://github.com/johngarcia73/Smart-History-Teacher`

3. Arquitectura del sistema

3.1. Diseño general multiagente

La arquitectura sigue un modelo híbrido **RAG (Retrieval-Augmented Generation)** coordinado por siete agentes especializados (Figura 1). La comunicación se realiza mediante mensajería **XMPP** usando el framework **SPADE**, con un patrón publish/subscribe y filtros de mensajes. Los agentes operan de forma autónoma pero colaborativa, permitiendo escalabilidad y tolerancia a fallos.

3.2. Integración del modelo RAG

El sistema implementa una capa dual:

- **Capa de Retrieval**: Combinación del agente Buscador (índice FAISS + ontología histórica) y el agente Crawler (búsqueda web en tiempo real)
- **Capa de Generation**: Agente Gestor de Prompts que construye instrucciones dinámicas basadas en el perfil del usuario

La ontología histórica permite el **aumento contextual** de consultas mediante expansión de términos y relaciones semánticas.

4. Explicación de la solución implementada

4.1. Agentes y responsabilidades

Agente Moodle (Interfaz de usuario)

- Monitoriza mensajes nuevos en Moodle cada 30 segundos
- Implementa bloqueo temporal de usuarios durante procesamiento
- Gestiona entrega final de respuestas

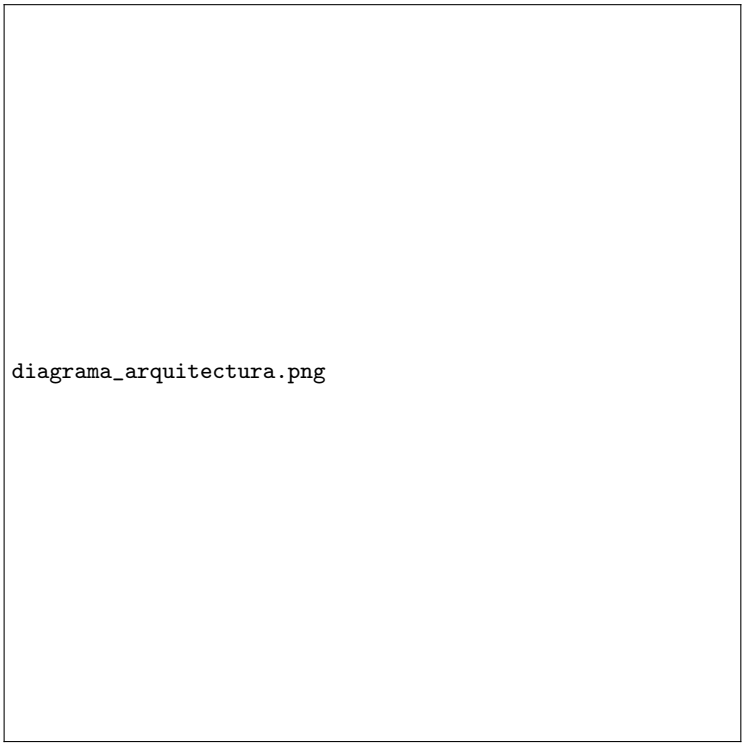

Figura 1. Arquitectura del sistema multiagente

Agente Analizador de Personalidad (Perfilado cognitivo)

- Analiza estilo de comunicación: humor (0.0-1.0), formalidad (0.0-1.0), preferencias de respuesta
- Genera perfiles mediante prompts estructurados:

```
prompt = f"""[INST]ANALIZAR:
- humor_score: 0.0-1.0
- formality_level: 0.0-1.0
- response_style: [direct|narrative|technical][/INST]"""
```

Agente Gestor de Perfiles (Modelado de usuario) Mantiene perfil dinámico del estudiante:

```
perfil_estudiante = {
  "preferences": {
    "history_specific": {
      "historiographical_approach": "social",
      "evidence_preference": ["primary_sources"]
    },
    "topics": {
      "preferred": ["Revoluci n Francesa"],
      "disliked": ["Guerras Mundiales"]
    }
  },
  "interaction_history": {
    "avg_engagement": 0.85,
    "preferred_response_types": ["narrative"]
  }
}
```

Agente Buscador (Recuperación contextual)

- Indexación semántica con FAISS y modelos *SentenceTransformers*
- Expansión de consultas mediante ontología histórica
- Integración de búsqueda híbrida (semántica + léxica)

Agente Evaluador (Validación de relevancia) Implementa algoritmo híbrido de evaluación:

```
def evaluate(candidates, query):
    # Normalizaci n FAISS
    faiss_scores = 1/(1 + distances)
    # C lculo BM25 con bonus por frases
    bm25_scores = bm25.get_scores(query) + phrase_bonus
    # Combinaci n adaptativa
    weights = get_adaptive_weights(query_type)
```

```
    final_score = weights["faiss"]*faiss_norm + weights["bm25
    "]*bm25_norm
    if max(final_score) < CONFIDENCE_THRESHOLD:
        trigger_scraping(query)
```

Agente Crawler (Búsqueda externa)

- Búsqueda en tiempo real con DuckDuckGo
- Extracción de fragmentos web con límite de calidad
- Activación selectiva cuando confianza local ¡0.7

Agente Gestor de Prompts (Generación adaptativa) Construye prompts personalizados:

```
[INST]Adapta segun:
- Estilo: {style}
- Formalidad: {formality_level}
- Temas preferidos: {preferred_topics}
- Enfoque historico: {historiographical_approach}
Contexto: {context}
Pregunta: {query}[/INST]
```

4.2. Flujos principales del sistema

Flujo de consulta típica

1. Usuario envía pregunta mediante Moodle
2. Agente Moodle bloquea usuario y envía consulta a Analizador de Personalidad
3. Analizador expande consulta usando perfil histórico
4. Buscador recupera fragmentos relevantes (FAISS + ontología)
5. Evaluador calcula confianza: si ¡0.7 activa Crawler
6. Gestor de Prompts genera respuesta personalizada
7. Agente Moodle desbloquea usuario y entrega respuesta

Flujo de actualización de perfil

1. Analizador de Personalidad detecta nuevos patrones
2. Gestor de Perfiles actualiza parámetros dinámicos
3. Propagación de nuevos 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:

- 58 eventos históricos con relaciones temporales y causales
- 32 personajes clave con roles e influencias
- 15 ubicaciones geopolíticas con contexto histórico
- 12 conceptos historiográficos interrelacionados

Ejemplo de expansión semántica:

```
"Revoluci n Francesa" ->
"Revoluci n Francesa (influy  en Revoluci n Americana)"
```

Agente Evaluador (Validación de relevancia) [...] El algoritmo implementa normalización robusta mediante función sigmoidea optimizada (Sección 7.1), garantizando estabilidad en rangos de confianza [...]

Agente Gestor de Prompts (Generación adaptativa) [...] La selección de estrategias de prompt se realiza mediante mapeo adaptativo basado en perfiles (Sección 7.2), priorizando robustness y personalización [...]

5. Consideraciones de implementación

5.1. Selección de tecnologías

- **Lenguaje Python**: Dominante en ecosistema IA/ML
- **SPADE**: Framework para agentes con soporte XMPP
- **FAISS**: Búsqueda semántica eficiente a gran escala
- **SentenceTransformers**: Modelos para embeddings de texto
- **spaCy**: Procesamiento de lenguaje natural en español

5.2. Procesamiento de lenguaje natural

- Modelo de análisis de personalidad mediante LLMs
- Generación adaptativa con ajuste dinámico de parámetros.

5.3. Modelo de representación de conocimiento

Arquitectura híbrida:

- **Grafo RDF**: Ontología histórica con relaciones semánticas
- **Índice FAISS**: Fragmentos textuales con metadatos contextuales
- **Perfiles JSON**: Modelo dinámico de preferencias de usuario

5.4. Mecanismos adicionales

- Timeouts configurables en interacciones entre agentes
- Reintentos para APIs externas (Moodle, LLMs)
- Balanceo de carga mediante XMPP federado

6. Justificación teórico-práctica

6.1. Arquitectura multiagente vs alternativas

- **Ventajas**: Escalabilidad natural, tolerancia a fallos, especialización funcional
- **Comparación**: Supera a arquitecturas monolíticas en flexibilidad y adaptabilidad

6.2. Selección del modelo RAG

- Ideal para dominios con conocimiento evolutivo (historia)
- Combina precisión factual (retrieval) con adaptabilidad (generación)
- Mitiga alucinaciones de LLMs mediante anclaje contextual

6.3. Enfoque de metaheurísticas aplicadas

- Algoritmo híbrido FAISS+BM25 con pesos adaptativos
- Optimización bayesiana de parámetros LLM
- Técnicas de normalización robusta para puntuaciones

7. Experimentos y Optimizaciones

7.1. Optimización de Función Sigmoidea para Relevancia

Objetivo: Ajustar la pendiente (parámetro a) para transformar puntajes BM25/FAISS al intervalo $[0, 1]$, mejorando la discriminación de relevancia.

Cuadro 1. Efecto de la pendiente a en valores transformados

Valor original para una consulta (verdadero positivo): 0.42

Pendiente (a)	**Valor transformado**
1	0.51
5	0.58
6-10	0.60

Conclusión: Un valor de $a \geq 6$ comprime los extremos de la distribución, estabilizando la salida en $\approx 0{,}60$ para a $= 10$. Esto optimiza la discriminación de relevancia en búsquedas al reducir la sensibilidad a outliers. Implementado en el Agente Evaluador como:

7.2. Evaluación de mejor estrategia de determinación de chunks

Objetivo: Determinar la efectividad de estrategias de chunketización local en base a cuánto mejoran la evaluación de la respuesta. Para un total de 30 queries, el promedio de evaluación del máximo score local fue de:

Por número de caracteres 0.51
Por número de tokens 0.58
Por número de oraciones 0.62
Por ventana deslizante 0.48

Por tanto, se utilizó la estrategia de dividir por oraciones.

7.3. Evaluación de Prompts para Perfiles de Usuario

Objetivo: Determinar la efectividad de estrategias de prompt engineering para diferentes perfiles cognitivos.
Metodología: 5 prompts distintos son aplicados a 10 usuarios con perfiles distintos bajo 5 queries distintas. El objetivo es obtener el prompt que en promedio obtiene una mejor evaluación con respecto a las respuestas de un LLM (Deepseek-R1) frente a las mismas queries de hechas por los mismos perfiles.

Cuadro 2. Resultados de efectividad por tipo de prompt

Prompt	Tipo	Puntaje Promedio	Observaciones
4	Colaborativo	0.67	Más equilibrado (mín: 0.59)
2	Narrativo	0.64	Óptimo para perfiles emocionales
5	Precisión	0.61	Destaca en perfiles técnicos
1	Estructurado	0.56	Menor consistencia
3	Analítico	0.58	Mínimo absoluto (0.42)

Hallazgos clave:
- **Máximo rendimiento:** 0.76 (Prompt 5 + perfil técnico)
- **Mínimo rendimiento:** 0.42 (Prompt 3 + perfil "pop")

Conclusiones implementadas:
1. *Prompt 4* como estrategia predeterminada por robustez (alto promedio + sin valores bajos)

8. Declaración autocrítica

8.1. Logros principales

- Modelado dinámico de perfil cognitivo-histórico con actualización en tiempo real
- Pipeline completo de personalización desde consulta hasta respuesta

8.2. Limitaciones identificadas

- Complejidad en mantenimiento y expansión de ontología histórica
- Ineficiencia en procesos de indexación de documentos (embedding)
- Curva de aprendizaje pronunciada para configuración SPADE

8.3. Propuestas de mejora

- **Módulo de retroalimentación**: Mecanismos explícitos de calificación de respuestas
- **Optimización de embeddings**: Procesamiento por lotes y incremental
- **Preprocesamiento avanzado**: Filtrado semántico de fragmentos web
- **Integración Moodle**: Creación automática de foros y recursos basados en interacciones
- **Personalización avanzada:** Extender el modelo de asignación de prompts a nuevos dominios educativos.

8.4. Trabajo futuro

- Extensión a dominios educativos adicionales (literatura, ciencias sociales)
- Implementación de razonamiento causal para análisis histórico
- Integración con bases de datos académicas (JSTOR, SciELO)
- Desarrollo de módulo multilingüe para educación histórica

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`