Informe Técnico
EVA – Evaluador Virtual Académico

Darío López Falcón

Facultad de Matemática y Computación (MATCOM)
Universidad de La Habana

10 de julio de 2025

Resumen

Este proyecto tiene como objetivo automatizar la evaluación de respuestas académicas abiertas mediante modelos de lenguaje (LLMs), reduciendo la carga de trabajo docente y mejorando la objetividad.

Se diseñó una aplicación web compuesta por un backend en Flask y un frontend en Next.js.

La evaluación se realiza a través de prompts personalizados enviados a modelos de lenguaje de Fireworks.ai.

El sistema permite a los profesores gestionar preguntas, ver resultados y obtener retroalimentación automática.

A pesar de su efectividad en preguntas cerradas, presenta desafíos en preguntas abiertas que requieren razonamiento complejo.

Objetivos

Objetivo general:

- Automatizar la evaluación de respuestas escritas utilizando modelos de lenguaje.

Objetivos específicos:

- Disminuir el tiempo de corrección de respuestas abiertas.
- Garantizar consistencia y objetividad en las calificaciones.
- Proporcionar retroalimentación automática a los estudiantes.
- Diseñar un sistema modular, extensible y fácil de desplegar.

1. Definición del problema

La corrección de respuestas abiertas suele ser lenta, subjetiva y difícil de estandarizar. Este proyecto busca una solución automatizada que mantenga coherencia en las calificaciones, permita escalabilidad y ofrezca retroalimentación inmediata al estudiante.

2. Propuesta de solución

Se desarrolló una aplicación web con dos módulos principales:

1. **Backend:** construido con Flask, se encarga de la evaluación automática mediante un modelo LLM y gestiona los datos.
2. **Frontend:** desarrollado con Next.js y TailwindCSS, permite a los docentes crear preguntas, filtrar resultados y visualizar evaluaciones.

3. Tecnologías utilizadas

Frontend

- Next.js (React + SSR)
- TailwindCSS
- TypeScript

Backend

- Flask + SQLAlchemy + Flask-Migrate
- Python 3.12+
- API de Fireworks.ai

4. Justificación de herramientas

Las tecnologías fueron seleccionadas por su eficiencia, comunidad activa y facilidad de integración. Flask ofrece una solución ligera para APIs; Next.js permite crear interfaces rápidas y modernas; Fireworks.ai provee LLMs de alta calidad.

5. Requisitos del sistema

Requisitos técnicos:

- Python 3.12+, Node.js, navegador moderno
- Conexión a internet y clave de API para Fireworks.ai

Requisitos funcionales:

- Crear, editar y evaluar preguntas académicas
- Filtrar resultados, visualizar puntuaciones y retroalimentación

6. Instalación y despliegue (Linux)

Backend:

```
git clone https://github.com/ArcanoxXx-01/EVA.git
cd EVA/backend
python3 -m venv env
source env/bin/activate
pip install -r requirements.txt
echo FIREWORKS_API_KEY=tu-api-key > .env
python run.py
```

Frontend:

```
cd ../frontend
npm install
npm run dev
```

7. Estructura del proyecto

La estructura básica del proyecto se organiza de la siguiente manera:

- `EVA/`
  - `backend/`
    - `app/` (contiene `models`, `routes`, `services`, `llm`)
    - `run.py`
    - `requirements.txt`
  - `frontend/`
    - `app/`, `public/`, `src/`
    - `package.json`, `tsconfig.json`, etc.

8. Diseño y arquitectura

La aplicación sigue una arquitectura desacoplada. El backend se comunica con el modelo LLM a través de una interfaz abstracta 'BaseLLM', lo cual permite cambiar de proveedor fácilmente y mantener el sistema escalable y modular.

9. Evaluación automática con Fireworks.ai

El backend genera un prompt estructurado con:

- Tema
- Tipo de pregunta
- Solución esperada

- Respuesta del estudiante
- Criterio de evaluación

El modelo devuelve una calificación numérica y una justificación textual.

10. Pruebas y validación

Las pruebas se centraron en:

- Preguntas de opción múltiple y verdadero/falso, con alta precisión en la evaluación.
- Verificación del correcto armado de prompts y retorno esperado del modelo.
- Validación funcional del frontend (filtros, navegación, modo oscuro).

11. Análisis del proyecto

Fortalezas

- Arquitectura modular y desacoplada.
- Evaluación funcional de tipos de preguntas cerradas.
- Interfaz moderna e intuitiva.

Limitaciones

- No se persisten las evaluaciones generadas.
- Desempeño variable en preguntas abiertas.
- Ausencia de comparación sistemática con evaluadores humanos.

12. Trabajos a futuro

A partir del desarrollo actual del sistema, se identifican múltiples líneas de trabajo que permitirían ampliar su funcionalidad, robustez y utilidad en entornos reales:

- **Persistencia de evaluaciones:** actualmente las respuestas generadas por el modelo no se almacenan. Guardar estos datos permitiría análisis estadísticos, validación posterior y trazabilidad de evaluaciones.
- **Historial por estudiante y problema:** agregar una funcionalidad que almacene los resultados de cada intento por estudiante ayudaría a monitorear la evolución del aprendizaje y detectar inconsistencias o mejoras.
- **Refinamiento del prompt:** mejorar el diseño del prompt utilizado con el modelo, empleando estrategias como few-shot learning, ejemplos concretos o ajuste dinámico según el tipo de pregunta.

- **Autenticación y roles de usuario:** implementar un sistema de inicio de sesión con permisos diferenciados (docente, administrador, etc.) para mejorar la seguridad y escalabilidad del sistema.
- **Soporte para múltiples modelos:** permitir el uso de distintos proveedores de LLMs (OpenAI, Cohere, Mistral...) usando la interfaz común ya existente (‘BaseLLM‘), de modo que el sistema sea más flexible y se puedan comparar diferentes motores de evaluación.
- **Visualización y exportación de resultados:** incluir reportes automáticos en PDF, Excel o gráficos de rendimiento por estudiante, problema o criterio de evaluación.
- **Feedback visual:** mostrar claramente el razonamiento generado por el modelo, indicando por qué una respuesta fue considerada correcta o incorrecta, lo que facilita la comprensión del estudiante.
- **Comparación con evaluaciones humanas:** realizar estudios de correlación entre evaluaciones hechas por el sistema y docentes reales, lo cual permitiría medir la precisión y confiabilidad del sistema en distintos contextos.
- **Uso de técnicas avanzadas de IA:** explorar el uso de *fine-tuning* sobre modelos base o técnicas de *retrieval-augmented generation* (RAG) para mejorar la calidad de las evaluaciones, especialmente en preguntas abiertas.
- **Sistema de colas y tareas asíncronas:** implementar una arquitectura con workers y procesamiento en segundo plano (por ejemplo, usando Celery o RQ) para manejar grandes volúmenes de evaluaciones sin afectar el rendimiento del servidor.

13. Conclusiones

El proyecto demuestra que es posible automatizar la evaluación académica con modelos LLM de forma efectiva en preguntas cerradas. La arquitectura desacoplada y el uso de una interfaz común para LLMs hacen que el sistema sea mantenible y extensible. Como trabajo futuro se propone mejorar la persistencia, refinar los prompts, y validar más rigurosamente la calidad de las respuestas del modelo.

Licencia