

Proyecto 2 — Multi-Agent Climate System (Documento Completo)

Desglose del Proyecto

El Multi-Agent Climate System es una plataforma avanzada diseñada para recolectar, procesar, analizar y visualizar narrativas climáticas provenientes de múltiples fuentes. El sistema integra scraping dinámico, APIs, procesamiento de lenguaje natural, modelos Transformers y agentes autónomos especializados para generar insights profundos sobre el discurso global respecto al cambio climático.

Fases Extendidas del Proyecto

FASE 1 — Recolección Masiva

- Scraping con Playwright.
- Scraping estático con BeautifulSoup.
- APIs: Twitter, Reddit, NewsAPI.
- Fuentes adicionales: RSS, blogs científicos, ONGs climáticas, medios internacionales.
- Validación, scheduling y logging.
- Almacenamiento en MongoDB.

FASE 2 — Preprocesamiento NLP

- Limpieza, tokenización, normalización.
- Embeddings: TF-IDF, Word2Vec, FastText, Sentence-BERT.
- Detección de idioma, lematización, stopwords personalizadas.
- NER preliminar.

FASE 3 — Modelado NLP

- Clasificación: SVM, Logistic Regression.
- Topic Modeling: LDA, BERTopic.
- Transformers: BERT, RoBERTa, DeBERTa.
- Análisis emocional, detección de sesgos, resúmenes automáticos.

FASE 4 — Sistema Multi-Agente

- Agente recolector.

- Agente analítico.
- Agente de clustering.
- Agente de resumen contextual.
- Agente ético.
- Agente de contradicción
- Agente histórico,
- Agente explorador.

FASE 5 — Visualización

- Dashboard en Streamlit.
- Gráficas interactivas, mapas narrativos, tendencias temporales.
- Wordclouds, redes semánticas.

Tecnologías por Fase

FASE 1:

- Playwright, BeautifulSoup, Trafilatura.
- Twitter API, Reddit API, NewsAPI.
- MongoDB, APScheduler, Prefect/Airflow.

FASE 2:

- spaCy, NLTK, Gensim.
- scikit-learn, FastText, SentenceTransformers.

FASE 3:

- scikit-learn, BERTopic, UMAP, HDBSCAN.
- HuggingFace Transformers.
- PyTorch / TensorFlow.

FASE 4:

- LangChain, FAISS.
- HuggingFace Summarization models.

- OpenAI / Llama / Mistral (opcional).

FASE 5:

- Streamlit, Plotly, Altair, PyVis.

Arquitectura Completa

Backend:

- Python, FastAPI, LangChain.
- MongoDB como base de datos principal.
- FAISS como vector store.

NLP Pipeline:

- spaCy / NLTK para limpieza.
- Gensim / SentenceTransformers para embeddings.
- Transformers (BERT/DeBERTa).

Orquestación:

- APScheduler para tareas periódicas.
- Prefect o Airflow para pipelines.

Frontend:

- Streamlit con visualizaciones interactivas.

Flujo de los Datos

1. Recolección:

Scraping y APIs → MongoDB RAW

2. Preprocesamiento:

Limpieza + Tokenización + Embeddings → MongoDB CLEAN

3. Modelado:

LDA / BERTopic / BERT → MongoDB CURATED

4. Multi-Agente:

Análisis + Resumen + Ética → Insights

5. Visualización:

Streamlit → Usuario Final

Componentes del Sistema

- Módulo de recolección (scrapers, APIs, RSS).
- Módulo de limpieza y normalización.
- Módulo de embeddings y análisis semántico.
- Módulo de modelado (topic modeling, clasificación).
- Arquitectura multi-agente para análisis avanzado.
- Dashboard interactivo.
- Base de datos MongoDB estructurada por niveles (RAW, CLEAN, CURATED).

Esquema Final para Implementarlo

1. Crear la estructura del repositorio.
2. Implementar recolección (scrapers, APIs).
3. Construir validaciones y scheduler.
4. Implementar pipeline de limpieza y embeddings.
5. Entrenar modelos NLP (clustering, clasificación, transformers).
6. Desarrollar agentes autónomos.
7. Integrar el vector store FAISS.
8. Construir dashboard en Streamlit.
9. Conectar todos los módulos mediante APIs.
10. Documentar completamente el sistema.