CONTEXT-AWARE CRS EVALUATION - TECHNICAL WALKTHROUGH

1. Repository layout

- dataset.py loads opendialkg_movie_data.json, normalizes genres, and builds attribute indexes (genre, actor, director, writer, language, year, plot keywords).
- user_sim.py synthesizes human-like requests with staged detail probes, constraint shifts across all metadata, and optional LLM drift.
- system.py contains CRSSystem, which extracts constraints, queries dataset.recommend, and prompts local Ollama models (or mock replies) with contextual payloads.
- simulate.py orchestrates a 20 turn dialogue between the UserSimulator and CRSSystem, persisting each transcript under logs/.
- evaluate.py computes every metric (segmentation, recovery, interference, CAS, LLM judge), using LDA topics when gensim models are available and the new TF-IDF embedder fallback when transformer weights cannot load.
- analyze_results.py digests results.jsonl into charts (bars, radar, deviation steps, topic-count density, per-metric distributions).

2. Simulation and logging

- batch_run.py clears prior runs, then loops over each model in config.LLM_SYSTEMS.
- simulate.run_simulation builds alternating USER/SYSTEM turns, storing the CRS constraint state alongside raw text for downstream analysis.
- Each session is saved as logs/<model>_session_<id>.json with model metadata to keep evaluation deterministic.

3. Constraint-aware CRS

- CRSSystem._update_constraints inspects every user turn for genre, year, actor, director, language, writer, explicit title, and plot cues (via find_plot_keywords_in_text).
- Recommendations are fetched through dataset.recommend, which intersects all active constraints against the indexed MOVIE_DB.
- The reply prompt injects summarized constraints plus top matches so the LLM response stays grounded.

4. Evaluation pipeline

- detect_user_segments groups USER turns via embedding_similarity and topic heuristics. The new embedding backend first tries sentence-transformers (when USE_TRANSFORMER_EMBEDDER=1 and torch works) and otherwise instantiates a TfidfSentenceEmbedder seeded by all movie plots.
- Per shift we compute recovery delay, interference, and segment topics; cross_coherence and context_retention reuse the same similarity primitive.
- ensure_lda_model loads train_lda.py outputs (models/lda.model, models/lda.dict) and falls back gracefully if gensim/nltk data are missing.
- Results per conversation are appended to results.jsonl together with llm_judge scores.

5. Visualization and summaries

- analyze_results.py now stabilizes the detail column, draws KDE distributions per metric, rebuilds CAS, radar, stacked, and deviation plots, and exports judge CSVs per model.
- model_metrics.csv captures aggregated numeric means, while model_metrics_show.csv offers a rounded snapshot for quick sharing.

6. Execution checklist
- Activate the venv, ensure USE_TRANSFORMER_EMBEDDER is toggled to 0 unless torch+numpy are rebuilt, run python batch_run.py to simulate+evaluate, then python analyze_results.py to refresh figures.
- For experimentation, python finetune_embeddings.py can train a TSDAE model once the transformer stack is available and config.EMBEDDING_MODEL_NAME points to the finetuned directory.