Code–base Overview

Below is a brief description of every document in the repository together with the *single element* (function or class) that is most critical inside that file.

1. **prompts.py**
   Generates all system / user prompts used by the red–team pipeline. **Key** – `high_level_policy_prompt`: produces the JSON template that instructs a helper-LLM to create jailbreak options and a hierarchical policy.

2. **main.py**
   Program entry-point. Loads models, iterates over HarmBench behaviours, spawns attacks, judges results, stores outputs. **Key** – `main(args)`: coordinates dataset I/O, attack generation, scoring and early-stop logic.

3. **LM_util_sonnet.py**
   Unified loader/wrapper for target and helper LLMs; also contains JSON-extraction helpers. **Key** – `class TargetLM` (and sibling `PolicyLM`): converts raw models into a simple `get_response` API.

4. **lib_utils.py** (Optional)
   Tiny persistence layer: an in-memory FAISS vector store that retains successful jailbreak policies for retrieval. **Key** – `save_policy_lib`: writes a new policy/option pair into the vector store.

5. **language_models_sonnet.py**
   Large adapter that standardises calls to OpenAI, Anthropic, Vertex AI, HuggingFace, etc. **Key** – `class GPT` (and analogues): wraps each vendor API with automatic retry and a uniform `batched_generate` interface.

6. **evaluation_harmbench.ipynb**
   Post-run notebook: loads saved CSVs, computes success metrics and draws comparison bar charts. **Key** – final plotting cell that renders the `matplotlib` grouped-bar figure.

7. **config.py**
   Central place for model checkpoints and global hyper-parameters (temperatures, top-p, local paths). **Key** – the constant block itself (e.g. `LLAMA_PATH`, `ATTACK_TEMP`) imported throughout.

8. **attacker_sonnet.py**
   Implements the attacker LLM that converts a jailbreak template into a concrete "`"new_prompt"`" JSON. **Key** – `AttackLM.get_attack`: regenerates until a syntactically valid attack prompt is returned.

9. **reward_helper.py**
   Defines judge models that grade each response; supports GPT-based judge or dummy. **Key** – `class GPTJudge`: builds the system prompt, calls the OpenAI judge model, parses the `[[score]]` bracket into an integer reward.