Emoji Remover (Demojifier)

# Preprocess

Main function: `emoji_semantic_clean(text)`

Steps:

- Normalize text (remove special Unicode characters like ZWJ, variation selectors).
- Detect and collapse repeated emojis (😂😂 → 😂).
- Map emojis to words using `EMOJI_MAP` and `emoji.demojize()`.
- Handle emoji combos (🔥🔥 → absolutely awesome).
- Clean up extra spaces and elongated letters (“soooo” → “soo”).
- Lowercase everything for consistency.
- If text is only emojis → return a simple meaning like `(love)` or blank.
- Remove duplicate or redundant emoji meanings if nearby words already express them (e.g., “funny 😂” → “funny”).
- Collapse duplicates and add mild emphasis (“😂😂” → “(laughing)!!”).
- Remove decorative emojis like `(sparkles)` if text already has meaningful words.
- Final punctuation cleanup (`!!!` → `!!`).
- Output looks like:
  👉 `"that movie was (hilarious)!!"`

# LLM Processing

Function: `emoji_to_meaning(client, text, model)`

Steps:

- If no emojis → return original text as-is.
- Otherwise, call LLM with a structured prompt:
  → “Convert emojis to plain English meaning in JSON {response: "..."}”
- Parse model response safely (expects JSON).
- If model returns wrong format or single “0/1” → fallback to deterministic conversion (`_to_words_no_parens`).
- If emojis remain → clean them again.
- Example output:
  👉 `{"response": "That’s hilarious!"}`

# Validator

Function: `evaluate_consistency_zero_one(client, standard_text, llm_text)`

Purpose:

Checks if LLM output means the same as the rule-based output.

Process:

- LLM judge outputs **“1”** if meanings match, **“0”** if not.
- Simple yes/no, no extra text allowed.
- If verdict can’t be parsed → treated as failed.

# Decision Making

Function: `demojify(text, client, model)`

Workflow:

1. Run **standard** demojifier → get `(meaning)` version.
2. Run **LLM** demojifier → get plain text version.
3. Validate:
    - If **validator = 1** → use **LLM output** (accepted).
    - If **validator = 0** → use **standard output** (rejected).
    - If validator fails or LLM errors → fallback to **standard output**.

Final Output Includes:

- `final_text` → chosen cleaned text
- `source` → “llm” or “standard”
- `reason` → e.g. `llm_valid`, `validator_error`, `llm_error`