Package 'llmclean'

June 9, 2026

Type Package

Title LLM-Assisted Data Cleaning with Multi-Provider Support

Version 0.1.1

Date 2026-06-01

Maintainer Sadikul Islam <sadikul.islamiasri@gmail.com>

Description Detects and suggests fixes for semantic inconsistencies in data frames by calling large language models (LLMs) through a unified, provider-agnostic interface. Supported providers include 'OpenAI' ('GPT-4o', 'GPT-4o-mini') <https://platform.openai.com>, 'Anthropic' ('Claude') <https://www.anthropic.com>, 'Google' ('Gemini') <https://ai.google.dev>, 'Groq' (free-tier 'LLaMA' and 'Mixtral') <https://groq.com>, and local 'Ollama' models <https://ollama.com>. The package identifies issues that rule-based tools cannot detect: abbreviation variants, typographic errors, case inconsistencies, and malformed values. Results are returned as tidy data frames with column, row index, detected value, issue type, suggested fix, and confidence score. An offline fallback using statistical and fuzzy-matching methods is provided for use without any application programming interface (API) key. Interactive fix application with human review is supported via 'apply_fixes()'. Methods follow de Jonge and van der Loo (2013) <https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf> and Chaudhuri et al. (2003) <doi:10.1145/872757.872796>.

License GPL-3

Depends R (>= 4.1.0)

Imports stats, utils, dplyr (>= 1.0.0), rlang (>= 1.0.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), httr2 (>= 1.0.0), jsonlite (>= 1.8.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Sadikul Islam [aut, cre] (ORCID:
<https://orcid.org/0000-0003-2924-7122>),
Rajesh Kaushal [aut]

Repository CRAN

Date/Publication 2026-06-09 14:20:02 UTC

Contents

`llmclean-package` *llmclean: LLM-Assisted Data Cleaning with Multi-Provider Support*

Description

Detects and suggests fixes for semantic inconsistencies in data frames by calling large language models through a unified, provider-agnostic interface.

The problem

Traditional data cleaning tools such as `janitor`, `validate`, and `pointblank` are rule-based: they catch type mismatches, range violations, and schema errors. They cannot understand *meaning*.

An LLM, by contrast, understands that `"NYC"`, `"New York"`, `"new york"`, and `"New Yrok"` all refer to the same city. It recognises `"jane@gmail.com"` as a malformed email, and `"actve"` as a typo for `"active"`. These semantic issues are the hardest to catch and the most damaging to downstream analyses.

Supported providers

OpenAI GPT-4o, GPT-4o-mini (API key required). The most widely used commercial LLM with mature tooling.

Anthropic Claude Sonnet, Claude Haiku (API key required). Large context window; strong instruction following.

Google Gemini 2.0 Flash, Gemini 1.5 Pro (API key required). Free tier available.

Groq LLaMA 3.1, Mixtral (free-tier API key). Fastest inference; suitable for large data frames.

Ollama Any local model (llama3, mistral, phi3). Fully offline; no API key; complete data privacy.

Workflow

1. **Configure provider**: `set_llm_provider()`
2. **Detect semantic issues**: `detect_issues()`
3. **Get suggested fixes**: `suggest_fixes()`
4. **Apply fixes interactively**: `apply_fixes()`
5. **Summary report**: `llmclean_report()`

Offline fallback

`offline_detect()` provides statistical and fuzzy-matching detection without any LLM call, using Levenshtein string distances, frequency analysis, and regex pattern matching. Use this when no API key is available or for data privacy reasons.

References

de Jonge, E. and van der Loo, M. (2013). An introduction to data cleaning with R. *Statistics Netherlands Discussion Paper*, The Hague. https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

Chaudhuri, S., Ganjam, K., Ganti, V. and Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, 313–324. doi:10.1145/872757.872796

van der Loo, M.P.J. and de Jonge, E. (2018). *Statistical Data Cleaning with Applications in R*. John Wiley & Sons, Chichester. doi:10.1002/9781118897126

OpenAI (2024). GPT-4 Technical Report. *arXiv preprint* arXiv:2303.08774. https://arxiv.org/abs/2303.08774

Author(s)

Maintainer: Sadikul Islam <sadikul.islamiasri@gmail.com> (ORCID)

Authors:

- Rajesh Kaushal

apply_fixes *Apply Suggested Fixes to a Data Frame*

Description

Applies the suggestions from detect_issues() or suggest_fixes() to the original data frame, either automatically or with interactive human review of each fix.

Usage

```
apply_fixes(df, issues, confirm = TRUE, min_confidence = 0.7, dry_run = FALSE)
```

Arguments

df	A data.frame. The original (uncleaned) data.
issues	A tibble of detected issues from detect_issues() or offline_detect().
confirm	Logical. If TRUE (default), show each proposed fix and ask the user to accept, reject, or enter a custom value. If FALSE, all fixes above min_confidence are applied automatically without prompting.
min_confidence	Numeric (0–1). Only apply fixes with confidence at or above this threshold. Default 0.70. Issues with suggestion = "REVIEW" are never applied automatically.
dry_run	Logical. If TRUE, return a summary of what would be changed without actually modifying the data. Default FALSE.

Details

The function creates a copy of df, then iterates over each row of issues in order of descending confidence. For each issue:

- If confirm = FALSE and confidence >= min_confidence and suggestion != "REVIEW": the value at df[row_index, column] is replaced with suggestion.
- If confirm = TRUE: the user is presented with the current value, the suggested fix, and the context, then prompted to accept (y), skip (n), or type a custom replacement.

A "_applied" attribute is attached to the returned data frame recording which fixes were applied, for use by llmclean_report().

Value

A data.frame with the same structure as df but with accepted fixes applied. The original df is not modified.

References

van der Loo, M.P.J. and de Jonge, E. (2018). *Statistical Data Cleaning with Applications in R*. John Wiley & Sons. doi:10.1002/9781118897126

See Also

detect_issues, suggest_fixes, llmclean_report

Examples

```
set_llm_provider("offline", verbose = FALSE)
data(messy_employees)
issues <- offline_detect(messy_employees)

# Non-interactive: apply high-confidence fixes automatically
df_clean <- apply_fixes(messy_employees, issues,
                        confirm = FALSE, min_confidence = 0.85)
head(df_clean)

# Dry run: see what would change
plan <- apply_fixes(messy_employees, issues, dry_run = TRUE)
plan
```

---

detect_issues *Detect Semantic Inconsistencies in a Data Frame Using an LLM*

Description

Sends a compact representation of the data frame to the configured LLM provider and requests detection of semantic inconsistencies: typographic errors, abbreviation variants, case inconsistencies, malformed values, cross-field contradictions, and implausible entries. Returns a tidy tibble of detected issues with location, type, and confidence.

Usage

```
detect_issues(
  df,
  columns = NULL,
  sample_n = NULL,
  issue_types = c("typo", "case", "abbreviation", "format", "outlier", "contradiction",
    "duplicate"),
  context = NULL,
  max_tokens = 2000L,
  verbose = TRUE
)
```

Arguments

df	A data.frame or tibble. The data to inspect.
columns	Character vector or NULL. Column names to inspect. If NULL (default), all character and factor columns are inspected plus numeric columns for outlier detection.

sample_n	Integer or NULL. Maximum number of rows to send to the LLM. If NULL (default), all rows are sent. For large data frames, set this to e.g. 200L to reduce token usage.
issue_types	Character vector. Types of issues to look for. Any subset of: "typo" (spelling errors), "case" (capitalisation inconsistency), "abbreviation" (variant forms of same entity), "format" (malformed values such as emails, dates, phone numbers), "outlier" (implausible numeric values), "contradiction" (cross-field logical conflict), "duplicate" (near-duplicate rows or values). Default: all types.
context	Character or NULL. Optional plain-English description of the data to help the LLM interpret values (e.g. "Employee records for a mid-size hospital. Age is in years. Status can be active or inactive.").
max_tokens	Integer. Maximum tokens in the LLM response. Default 2000L.
verbose	Logical. Print progress messages. Default TRUE.

Details

The function constructs a structured prompt containing:

1. Column names and inferred types.
2. The first sample_n rows in CSV-like format.
3. Instructions requesting JSON output with one object per issue.

The LLM is instructed to return **only** a JSON array with fields: column, row_index (1-based), value, issue_type, explanation, suggestion, and confidence (0–1).

If the LLM response cannot be parsed as valid JSON, the raw response is returned with a warning and offline_detect() is called as fallback.

Token usage note: Sending 100 rows of a 10-column data frame uses approximately 1,500–3,000 tokens. At gpt-4o-mini pricing (~$0.15/M input tokens as of 2025), this costs less than US$0.001.

Value

A tibble with one row per detected issue and columns:

column Column name where the issue was found.

row_index Row number (1-based).

value The problematic value as a character string.

issue_type One of the issue_types categories.

explanation Human-readable explanation of the problem.

suggestion Suggested corrected value.

confidence Numeric 0–1. LLM-assigned confidence.

provider The LLM provider used.

model The specific model used.

Returns a zero-row tibble with the same columns if no issues are found.

References

de Jonge, E. and van der Loo, M. (2013). An introduction to data cleaning with R. *Statistics Netherlands Discussion Paper*. https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

Chaudhuri, S., Ganjam, K., Ganti, V. and Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. *Proc. 2003 ACM SIGMOD*, 313–324. doi:10.1145/872757.872796

See Also

set_llm_provider, suggest_fixes, apply_fixes, offline_detect

Examples

```
# Offline detection (no API key needed)
set_llm_provider("offline", verbose = FALSE)

data(messy_employees)
issues <- detect_issues(messy_employees)
issues

# With LLM (requires valid API key)
## Not run:
set_llm_provider("openai", model = "gpt-4o-mini")
issues <- detect_issues(messy_employees,
                        context = "Employee records. Status: active/inactive.")
issues

# Groq free tier
set_llm_provider("groq", model = "llama-3.1-8b-instant")
issues <- detect_issues(messy_employees, sample_n = 50L)

## End(Not run)
```

---

get_llm_provider *Get Current LLM Provider Configuration*

Description

Returns the current LLM provider, model, and base URL configured by set_llm_provider().

Usage

```
get_llm_provider(show_key = FALSE)
```

Arguments

show_key	Logical. If TRUE, include the API key (truncated) in the output. Default FALSE.

Value

A named list with elements `provider`, `model`, `base_url`, and optionally `api_key`.

See Also

`set_llm_provider`

Examples

```
set_llm_provider("offline", verbose = FALSE)
get_llm_provider()
```

---

llmclean_report *Generate a Summary Report of LLM-Assisted Data Cleaning*

Description

Produces a tidy summary of detected issues and applied fixes, grouped by column and issue type. Suitable for inclusion in a data-quality audit log or reproducible report. Optionally prints a formatted table to the console.

Usage

```
llmclean_report(df_original, df_cleaned, issues, print = TRUE)
```

Arguments

`df_original`	A `data.frame`. The original (uncleaned) data.
`df_cleaned`	A `data.frame`. The cleaned data returned by `apply_fixes()`.
`issues`	A `tibble` of detected issues from `detect_issues()` or `offline_detect()`.
`print`	Logical. If `TRUE` (default), prints a formatted summary table to the console.

Details

The report computes:

- Number of issues detected per column and issue type.
- Number of fixes applied vs skipped.
- Cell-level change summary (original value -> corrected value).
- Provider and model used.
- Data dimensions before and after cleaning.

The applied-fix provenance is read from the `"_applied"` attribute set by `apply_fixes()`. If the attribute is absent (e.g. when `df_cleaned` was not produced by `apply_fixes()`), the report uses the full `issues` tibble.

Value

Invisibly returns a named list with three elements: `summary` (tibble: column / issue_type / n_detected / n_applied), `changes` (tibble: column / row_index / original / corrected), `metadata` (list: provider, model, n_total, n_applied, n_skipped).

References

de Jonge, E. and van der Loo, M. (2013). An introduction to data cleaning with R. *Statistics Netherlands Discussion Paper*. https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

See Also

`detect_issues`, `apply_fixes`

Examples

```
set_llm_provider("offline", verbose = FALSE)
data(messy_employees)
issues   <- offline_detect(messy_employees)
df_clean <- apply_fixes(messy_employees, issues,
                        confirm = FALSE, min_confidence = 0.85)
rpt <- llmclean_report(messy_employees, df_clean, issues)
rpt$summary
```

---

messy_employees *Hypothetical Messy Employee Records Dataset*

Description

A hypothetical data frame of 20 employee records containing deliberate data quality issues across all common inconsistency types: capitalisation variants, typos, duplicate entries, malformed email addresses, out-of-range numeric values, and cross-field redundancies. Designed to illustrate the full range of issues detectable by `detect_issues()` and `offline_detect()`.

Usage

```
messy_employees
```

Format

A `data.frame` with 20 rows and 8 variables:

emp_id Integer. Unique employee identifier (1–20).

name Character. Employee full name. Contains mixed-case inconsistencies (e.g. `"Alice Johnson"` vs `"alice johnson"` vs `"ALICE JOHNSON"`).

department Character. Department name. Contains case variants (`"Finance"` vs `"finance"`), abbreviations (`"IT"` vs `"I.T."`), synonym variants (`"HR"` vs `"Human Resources"`), and a typo (`"Finanace"`).

email Character. Email address. Contains malformed values: double `@`, missing TLD, and leading dot.

age Integer. Age in years. Contains two impossible values: `-5` and `150`.

salary Numeric. Annual salary in USD. Contains a data entry error (`999999`) as an implausible outlier.

status Character. Employment status (`active` / `inactive`). Contains case variants and two typos (`"actve"`).

hire_date Character. Hire date. Contains a mixed-format date (`"2015/07/22"` vs `"2018-03-15"` ISO-8601 format).

Details

All data are entirely hypothetical and generated for illustrative purposes. The inconsistency types are based on the taxonomy in de Jonge and van der Loo (2013) and reflect common real-world data entry errors documented in Muller and Freytag (2003).

Source

Hypothetical data generated for illustration. See `data-raw/generate_datasets.R`.

References

de Jonge, E. and van der Loo, M. (2013). An introduction to data cleaning with R. *Statistics Netherlands Discussion Paper*. https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

Muller, H. and Freytag, J.C. (2003). Problems, methods, and challenges in comprehensive data cleansing. *Technical Report, Humboldt University Berlin*, HUB-IB-164.

See Also

`messy_survey`, `detect_issues`, `offline_detect`

Examples

```
data(messy_employees)
str(messy_employees)

# Quick overview of known issues
table(messy_employees$status)      # case inconsistency
table(messy_employees$department)  # variant forms
messy_employees$age[messy_employees$age < 0 | messy_employees$age > 100]
```

`messy_survey` *Hypothetical Messy Survey Response Dataset*

Description

A hypothetical survey data frame of 15 respondents from 5 countries containing systematic data quality issues typical of free-text survey responses: country name variants (`"USA"` vs `"United States"`), satisfaction rating inconsistencies, spelling errors, and numeric outliers.

Usage

```
messy_survey
```

Format

A `data.frame` with 15 rows and 5 variables:

respondent_id Integer. Unique respondent identifier (1–15).

country Character. Respondent country. Contains abbreviations (`"USA"`, `"UK"`), synonyms (`"Deutschland"` for `"Germany"`), case variants, and a typo (`"Japn"`).

satisfaction Character. Satisfaction rating (5-point Likert scale). Contains case inconsistencies and typos (`"Satisified"`, `"Nutral"`, `"Very Dissatified"`).

age_group Character. Age group bracket. Clean column for comparison.

income_usd Numeric. Reported annual income in USD. Contains a negative value (`-500`) and an implausible outlier (`999999`).

Details

All data are entirely hypothetical. Survey inconsistency patterns are inspired by common free-text standardisation challenges described in Chaudhuri et al. (2003).

Source

Hypothetical data generated for illustration. See `data-raw/generate_datasets.R`.

References

Chaudhuri, S., Ganjam, K., Ganti, V. and Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. *Proc. 2003 ACM SIGMOD*, 313–324. doi:10.1145/872757.872796

See Also

`messy_employees`, `detect_issues`

Examples

```
data(messy_survey)
str(messy_survey)
table(messy_survey$country)       # variant forms
table(messy_survey$satisfaction)  # case + typos
```

`offline_detect` *Offline Detection of Data Inconsistencies Without an LLM*

Description

Detects data quality issues using statistical and fuzzy-matching methods, requiring no LLM API call or internet connection. Uses Levenshtein string distances for near-duplicate category detection, frequency analysis for rare/singleton values, regex patterns for format validation, and IQR-based outlier detection for numeric variables.

Usage

```
offline_detect(
  df,
  columns = NULL,
 issue_types = c("typo", "case", "abbreviation", "format", "outlier", "duplicate"),
  max_edit_distance = 2L,
  min_freq = 2L,
  outlier_iqr_mult = 3
)
```

Arguments

`df`	A `data.frame` or `tibble`.
`columns`	Character vector or `NULL`. Columns to inspect. If `NULL`, all character, factor, and numeric columns are included.
`issue_types`	Character vector. Subset of `"typo"`, `"case"`, `"abbreviation"`, `"format"`, `"outlier"`, `"duplicate"`. Default: all.
`max_edit_distance`	Integer. Maximum Levenshtein edit distance between two values to flag as potential near-duplicates (typos or abbreviations). Default `2L`.
`min_freq`	Integer. Values appearing fewer than this many times in a column are flagged as potentially erroneous singletons. Default `2L`.
`outlier_iqr_mult`	Numeric. Multiplier for the IQR-based outlier fence: values outside `[Q1 - k*IQR, Q3 + k*IQR]` are flagged. Default `3.0` (Tukey outer fence).

Details

Methods used:

Typo / near-duplicate detection Levenshtein edit distance (Levenshtein, 1966) between all pairs of unique values in each column. Pairs within `max_edit_distance` edits but with different capitalisation-normalised forms are flagged as potential typos. Implemented via `utils::adist()`.

Case inconsistency Detects columns where the same word appears in multiple capitalisation forms (e.g. `"active"`, `"Active"`, `"ACTIVE"`).

Format validation Regex patterns for email addresses, phone numbers (international), URLs, and ISO dates. Any value not matching the majority pattern is flagged.

Numeric outliers Modified Tukey outer fence: flags values beyond `Q1 - k*IQR` or `Q3 + k*IQR` where k = `outlier_iqr_mult`.

Singleton detection Values appearing only once in a column with fewer than 10 unique values are flagged (likely erroneous entries).

The offline detector cannot catch *semantic* issues (e.g. knowing that `"NYC"` and `"New York"` are the same city) — that requires an LLM. Use `detect_issues()` with a configured LLM provider for full semantic detection.

Value

A `tibble` with the same structure as `detect_issues()`: columns `column`, `row_index`, `value`, `issue_type`, `explanation`, `suggestion`, `confidence`. The `provider` and `model` columns are set to `"offline"` and `"statistical"` respectively.

References

Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8), 707–710.

Tukey, J.W. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, MA. ISBN: 978-0-201-07616-5.

Chaudhuri, S., et al. (2003). Robust and efficient fuzzy match for online data cleaning. *Proc. 2003 ACM SIGMOD*, 313–324. doi:10.1145/872757.872796

See Also

`detect_issues`, `set_llm_provider`

Examples

```
data(messy_employees)
issues <- offline_detect(messy_employees)
issues

# Only look for case and format issues
offline_detect(messy_employees, issue_types = c("case", "format"))
```

`set_llm_provider` *Configure the LLM Provider for Data Cleaning*

Description

Sets the LLM provider, API key, and model to use for all subsequent `detect_issues()` and `suggest_fixes()` calls. Configuration is stored in a package-level environment and persists for the R session. Supports OpenAI, Anthropic (Claude), Google (Gemini), Groq, and local Ollama models.

Usage

```
set_llm_provider(
  provider = "offline",
  api_key = NULL,
  model = NULL,
  base_url = NULL,
  verbose = TRUE
)
```

Arguments

`provider`	Character. LLM provider name. One of: `"openai"`, `"anthropic"`, `"google"`, `"groq"`, `"ollama"`, `"offline"` (default; no LLM call).
`api_key`	Character or `NULL`. API key for the chosen provider. If `NULL` (default), the function looks for the appropriate environment variable: `OPENAI_API_KEY`, `ANTHROPIC_API_KEY`, `GOOGLE_API_KEY`, `GROQ_API_KEY`. Not required for `"ollama"` or `"offline"`.
`model`	Character or `NULL`. Model name. If `NULL` (default), a sensible default is chosen per provider: **openai** `"gpt-4o-mini"` (fast, cheap, capable) **anthropic** `"claude-haiku-4-5-20251001"` (fast, cheap) **google** `"gemini-2.0-flash"` (free tier available) **groq** `"llama-3.1-8b-instant"` (free tier) **ollama** `"llama3"` (local)
`base_url`	Character or `NULL`. Custom base URL for the API endpoint. Useful for self-hosted deployments or API proxies. If `NULL`, provider defaults are used.
`verbose`	Logical. If `TRUE`, prints a confirmation message. Default `TRUE`.

Details

Getting API keys (all free tiers available):

OpenAI https://platform.openai.com/api-keys

Anthropic https://console.anthropic.com/

Google Gemini https://aistudio.google.com/app/apikey

Groq (free) https://console.groq.com/keys

Ollama (local) https://ollama.com/ — run `ollama pull llama3`

Best practice: Store API keys in environment variables rather than in scripts. Add to your `.Renviron` file:

```
OPENAI_API_KEY=sk-...
ANTHROPIC_API_KEY=sk-ant-...
GROQ_API_KEY=gsk_...
GOOGLE_API_KEY=AIza...
```

Value

Invisibly returns a list with the configured provider settings.

References

OpenAI (2024). GPT-4 Technical Report. *arXiv preprint* arXiv:2303.08774. https://arxiv.org/abs/2303.08774

Anthropic (2024). Claude Model Overview. https://platform.claude.com/docs/en/docs/about-claude/models/overview

See Also

get_llm_provider, detect_issues

Examples

```
# Use offline mode (no API key needed)
set_llm_provider("offline")

# Configure OpenAI (reads key from env var OPENAI_API_KEY)
## Not run:
set_llm_provider("openai", model = "gpt-4o-mini")

# Configure Anthropic explicitly
set_llm_provider("anthropic",
                 api_key = Sys.getenv("ANTHROPIC_API_KEY"),
                 model   = "claude-haiku-4-5-20251001")

# Configure free Groq
set_llm_provider("groq", model = "llama-3.1-8b-instant")

# Local Ollama (no key needed)
set_llm_provider("ollama", model = "llama3")

## End(Not run)
```

suggest_fixes *Request Enriched Fix Suggestions for Detected Issues*

Description

Takes the issues tibble from `detect_issues()` or `offline_detect()` and optionally sends it back to the LLM with the full data context to obtain higher-quality, rank-ordered suggestions for each issue. Useful when the initial detection pass returned `suggestion = "REVIEW"` or low-confidence suggestions.

Usage

```
suggest_fixes(
  df,
  issues,
  n_alternatives = 2L,
  filter_confidence = 0.8,
  verbose = TRUE
)
```

Arguments

`df`	A `data.frame`. The original (uncleaned) data.
`issues`	A `tibble` returned by `detect_issues()` or `offline_detect()`.
`n_alternatives`	Integer. Number of alternative suggestions per issue (in addition to the primary). Default `2L`.
`filter_confidence`	Numeric (0–1). Only re-query issues with confidence below this threshold. Default `0.80`. Set to `1.0` to re-query all issues.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Details

If the current provider is `"offline"`, this function returns the input `issues` tibble unchanged with a message, since no LLM is available to enrich suggestions.

The function sends each low-confidence issue to the LLM along with the surrounding rows for context, and requests ranked alternatives. Results are merged back into the issues tibble, replacing the original `suggestion` with the top-ranked alternative and adding `alternatives` as a comma-separated string column.

Value

The input `issues` tibble with two additional columns: `alternatives` (comma-separated list of alternative suggestions) and `confidence_revised` (updated confidence after re-querying).

References

de Jonge, E. and van der Loo, M. (2013). An introduction to data cleaning with R. *Statistics Netherlands Discussion Paper*. https://cran.r-project.org/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

See Also

detect_issues, apply_fixes

Examples

```
set_llm_provider("offline", verbose = FALSE)
data(messy_employees)
issues <- offline_detect(messy_employees)

# Offline: suggest_fixes returns issues unchanged
enriched <- suggest_fixes(messy_employees, issues)
enriched

## Not run:
set_llm_provider("openai", model = "gpt-4o-mini")
issues   <- detect_issues(messy_employees)
enriched <- suggest_fixes(messy_employees, issues, n_alternatives = 3L)

## End(Not run)
```

Index