Package 'AutoGenAI'

September 12, 2026

Type Package

Title Adaptive Optimization of Prompts, Models and Generation Strategies

Version 0.1.0

Description Provides provider-agnostic tools for jointly comparing and optimizing prompts, language-model providers, and generation strategies for generative artificial intelligence workflows. Candidate configurations can be evaluated using user-supplied scoring functions, cost and latency measurements, robustness perturbations, Pareto-front screening, budget and latency constraints, prompt evolution, adaptive routing, self-consistency, and text-output ensembles. The core workflow is designed to run offline with deterministic mock providers, while external model application programming interfaces can be connected through user-defined provider functions. Evolutionary search concepts are described by Goldberg (1989, ISBN:0201157675), and multi-objective optimization concepts are related to Deb, Pratap, Agarwal and Meyarivan (2002) <doi:10.1109/4235.996017>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports jsonlite

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Leila Marvian Mashhad [aut, cre]

Maintainer Leila Marvian Mashhad <leila.marveian@gmail.com>

Repository CRAN

Date/Publication 2026-09-12 07:40:02 UTC

Contents

`AutoGenAI-package` *Adaptive Optimization of Prompts, Models and Generation Strategies*

Description

Provider-agnostic tools for comparing and optimizing generative AI configurations with offline reproducible workflows.

Details

The package focuses on joint prompt-model-strategy evaluation, Pareto screening, robustness testing, routing, ensembles, and deployment constraints.

See Also

`optimize_ai`, `autogenai_example`

`ai_ensemble` *Combine repeated or multi-provider outputs*

Description

Provides exact-text consensus utilities for repeated generations and multiple providers.

Usage

```
consensus_vote(outputs, weights = NULL)
ai_ensemble(providers, prompt, input, params = list(),
  method = c("majority", "weighted"), weights = NULL)
self_consistency(provider, prompt, input, params = list(), samples = 3L)
```

Arguments

`outputs`	Text outputs to combine.
`weights`	Optional numeric voting weights.
`providers`	Provider list.
`prompt`, `input`, `params`	Generation inputs.
`method`	Majority or weighted voting.
`provider`	Single provider for repeated sampling.
`samples`	Number of repeated generations.

Value

Consensus text or a list containing consensus and member outputs.

`ai_provider` *Create generative AI providers*

Description

Creates provider objects. Real services are connected by user-supplied generation functions; mock providers require no network access.

Usage

```
ai_provider(
  name, generate, input_cost_per_1k = 0,
  output_cost_per_1k = 0, metadata = list()
)
mock_provider(
  name = "mock-balanced",
  profile = c("balanced", "strict", "cheap"), seed = 1L
)
```

Arguments

`name`	Provider name.
`generate`	Function with arguments `prompt`, `input`, and `params`; it must return one text value.
`input_cost_per_1k`, `output_cost_per_1k`	Estimated monetary cost per 1,000 tokens.
`metadata`	Optional provider metadata.
`profile`	Offline mock behavior profile.
`seed`	Integer seed stored with the mock provider.

Value

An object of class autogen_provider.

Examples

```
p <- mock_provider()
p
p$generate("Classify sentiment", "I love this", list(temperature = 0))
```

---

ai_task *Define an AutoGenAI task and scorer*

Description

Defines a provider-independent task. The default scorer performs case-insensitive exact matching after whitespace normalization.

Usage

```
ai_task(
  name, instruction, scorer = default_scorer,
  structured = c("none", "json"), metadata = list()
)
default_scorer(prediction, truth, ...)
```

Arguments

name	Task name.
instruction	Base task instruction.
scorer	Function mapping prediction and truth to a numeric quality score.
structured	Whether JSON output is expected.
metadata	Optional metadata.
prediction	Predicted text.
truth	Reference value.
...	Additional arguments ignored by the default scorer.

Value

ai_task() returns an autogen_task; default_scorer() returns a numeric score.

autogenai-methods *Methods for AutoGenAI optimization objects*

Description

Print, summarize, plot, and predict with an optimized AutoGenAI configuration.

Usage

```
## S3 method for class 'autogenai'
print(x, ...)
## S3 method for class 'autogenai'
summary(object, ...)
## S3 method for class 'autogenai'
plot(x, ...)
## S3 method for class 'autogenai'
predict(object, newdata, providers = NULL,
  abstain_threshold = NULL, ...)
```

Arguments

x, object	An object returned by optimize_ai().
...	Additional arguments passed to plotting methods where applicable.
newdata	A data frame with input or a character vector.
providers	Provider list containing the selected provider.
abstain_threshold	Optional minimum benchmark quality required for prediction.

Value

The print and plot methods return the object invisibly; summary returns a list; predict returns character outputs.

autogenai_example *Create an offline AutoGenAI example*

Description

Returns a small sentiment benchmark, a task, three deterministic mock providers, and candidate prompts.

Usage

```
autogenai_example()
```

Value

A list with task, data, providers, and prompts.

Examples

```
ex <- autogenai_example()
names(ex)
```

---

benchmark_ai *Benchmark generative AI configurations*

Description

Evaluates one or many prompt-provider-generation configurations while recording quality, latency, cost, token estimates, and success.

Usage

```
evaluate_config(task, data, provider, prompt, temperature = 0,
  strategy = c("single", "self_consistency"), samples = 1L, scorer = NULL)
benchmark_ai(task, data, providers, prompts, temperatures = 0,
  strategies = "single", samples = 1L, scorer = NULL)
optimize_generation(task, data, providers, prompts,
  temperatures = c(0, 0.2, 0.7),
 strategies = c("single", "self_consistency"), samples = c(1L, 3L), scorer = NULL)
```

Arguments

task	An autogen_task.
data	Data frame containing input and optionally truth.
provider, providers	One provider or a list of providers.
prompt, prompts	Prompt text or candidate prompts.
temperature, temperatures	Generation temperature values passed to providers.
strategy, strategies	Single generation or self-consistency.
samples	Number of repeated generations for self-consistency.
scorer	Optional scorer overriding the task scorer.

Value

evaluate_config() returns row-level results. benchmark_ai() and optimize_generation() return lists with raw and summarized results.

Examples

```
ex <- autogenai_example()
b <- benchmark_ai(ex$task, ex$data, ex$providers, ex$prompts[1:2])
head(b$summary)
```

`optimize_ai` *Optimize and select generative AI configurations*

Description

Jointly evaluates prompt, provider, and generation settings and supports utility, Pareto, budget, latency, and abstention decisions.

Usage

```
optimize_ai(task, data, providers, prompts, temperatures = c(0, 0.2),
  strategies = c("single", "self_consistency"), samples = c(1L, 3L),
  objective = c(quality = 0.55, robustness = 0.15, cost = 0.15,
  latency = 0.10, structured = 0.05), method = c("grid", "random"),
  max_configs = Inf, scorer = NULL, robustness_perturbations = NULL, seed = 1L)
pareto_ai(x, objectives = c(quality = "max", cost = "min", latency = "min"))
select_configuration(
  x,
  preference = c("balanced", "highest_quality", "cheapest", "fastest")
)
budget_optimize(x, max_cost, preference = c("balanced", "highest_quality"))
latency_optimize(x, max_latency, preference = c("balanced", "highest_quality"))
abstain_decision(score, threshold = 0.8)
```

Arguments

`task`, `data`, `providers`, `prompts`, `temperatures`, `strategies`, `samples`, `scorer`	See `benchmark_ai`.
`objective`	Named utility weights. Robustness is reserved for explicit stress testing and is not silently imputed.
`method`	Grid or random configuration search.
`max_configs`	Maximum configurations retained or randomly sampled.
`robustness_perturbations`	Optional perturbation types evaluated for every retained configuration and incorporated into the utility when a robustness weight is present.
`seed`	Random seed.
`x`	An `autogenai` object or compatible results data frame.
`objectives`	Named vector specifying whether each Pareto objective is maximized or minimized.
`preference`	Selection rule.

max_cost	Maximum mean cost per request.
max_latency	Maximum mean latency in seconds.
score	Reliability or quality score.
threshold	Acceptance threshold.

Value

optimize_ai() returns an object of class autogenai; selection functions return data frames or decision labels.

Examples

```
ex <- autogenai_example()
fit <- optimize_ai(ex$task, ex$data, ex$providers, ex$prompts,
                   temperatures = 0, strategies = "single")
fit
pareto_ai(fit)
select_configuration(fit)
```

---

prompt_candidates *Prompt search and cross-provider generalization*

Description

Creates, mutates, crosses, and evolves prompts, and computes stability-oriented prompt and provider indices.

Usage

```
prompt_candidates(base_prompt, n = 6L)
mutate_prompt(prompt, strength = 1L, seed = NULL)
crossover_prompt(prompt_a, prompt_b, seed = NULL)
evolve_prompt(task, data, providers, base_prompt, generations = 3L,
  population = 6L, temperature = 0, scorer = NULL, seed = 1L)
prompt_portability(results, lambda = 0.25)
model_generalization(results, category_col = "category", lambda = 0.25)
```

Arguments

base_prompt, prompt, prompt_a, prompt_b	Prompt text.
n	Number of candidate prompts.
strength	Number of mutation clauses added.
seed	Optional random seed.
task, data, providers, temperature, scorer	Evaluation inputs.

generations	Number of evolutionary generations.
population	Population size.
results	AutoGenAI results or a compatible data frame.
lambda	Penalty applied to cross-group standard deviation.
category_col	Column defining task categories for model generalization.

Value

Text, candidate vectors, an evolution object, or ranked data frames depending on the function.

`stress_test` *Stress-test generative AI inputs*

Description

Applies reproducible text perturbations and measures performance under changed inputs.

Usage

```
perturb_text(x, type = c("whitespace", "case", "typo", "distractor", "prefix"), seed = 1L)
stress_test(task, data, provider, prompt,
  perturbations = c("whitespace", "case", "typo", "distractor"),
  temperature = 0, scorer = NULL, seed = 1L)
robustness_score(x)
```

Arguments

x	Text or a stress-test result.
type	Perturbation type.
seed	Random seed.
task, data, provider, prompt, temperature, scorer	Evaluation inputs.
perturbations	Perturbations to apply.

Value

Perturbed text, a stress-test data frame, or a numeric robustness score.

`train_router` *Train and evaluate an adaptive provider router*

Description

Learns a lightweight one-vs-rest logistic router from text-shape features and empirical provider winners.

Usage

```
train_router(log, input_col = "input", provider_col = "provider",
  quality_col = "quality", task_id_col = NULL)
route_model(router, input, return_probabilities = FALSE)
update_router(router, new_log)
router_regret(log, chosen_col = "chosen_provider", provider_col = "provider",
  quality_col = "quality", task_id_col = "task_id")
```

Arguments

`log`	Evaluation log containing inputs, providers, and quality.
`input_col`, `provider_col`, `quality_col`, `task_id_col`	Column names.
`router`	Router returned by `train_router()`.
`input`	New text inputs.
`return_probabilities`	Return normalized one-vs-rest probabilities together with selected providers.
`new_log`	Additional routing observations.
`chosen_col`	Column containing the provider chosen by a router.

Value

A router, provider names, updated router, or regret data frame.

Index