

Package ‘asa’

May 21, 2026

Title AI Search Agent for Large-Scale Research Automation

Version 0.1.0

Author Connor Jerzak [aut, cre] (<<https://orcid.org/0000-0003-1914-8905>>)

Maintainer Connor Jerzak <connor.jerzak@gmail.com>

Description Provides an LLM-powered research agent for performing AI search tasks at large scales. Uses a ReAct (Reasoning + Acting) agent pattern with web search capabilities via DuckDuckGo and Wikipedia. Implements DeepAgent-style memory folding for context management. The agent is built on 'LangGraph' and supports multiple LLM backends including 'OpenAI', 'Groq', 'xAI', 'Gemini', 'Anthropic' (Claude), 'AWS Bedrock', 'OpenRouter', and 'Exo'.

URL <https://github.com/cjerzak/asa-software>

BugReports <https://github.com/cjerzak/asa-software/issues>

Depends R (>= 4.0.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Imports reticulate (>= 1.28),
jsonlite,
rlang,
digest,
yaml,
processx

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown,
future,
future.apply,
withr

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

SystemRequirements Python (>= 3.11), Conda, Tor (optional, for anonymous searching)

Contents

as.data.frame.asa_audit_result	3
as.data.frame.asa_enumerate_result	3
as.data.frame.asa_evaluation_result	4
as.data.frame.asa_result	4
asa_agent	5
asa_audit	5
asa_audit_result	7
asa_config	8
asa_enumerate	10
asa_enumerate_result	14
asa_evaluation_result	15
asa_response	16
asa_result	18
benchmark_from_df	19
benchmark_from_yaml	20
build_backend	20
build_prompt	22
check_backend	22
configure_search	23
configure_search_logging	24
configure_temporal	25
configure_tor_registry	26
create_benchmark	27
evaluate_agent	28
extract_agent_results	29
extract_search_snippets	30
extract_search_tiers	30
extract_urls	31
extract_wikipedia_content	32
get_agent	32
get_tor_ip	33
initialize_agent	33
is_tor_running	37
list_benchmarks	37
load_benchmark	38
print.asa_agent	38
print.asa_audit_result	39
print.asa_benchmark	39
print.asa_config	40
print.asa_enumerate_result	40
print.asa_evaluation_result	41
print.asa_response	41
print.asa_result	42
print.asa_search	42
print.asa_temporal	43
print.asa_tor	43
process_outputs	44
reset_agent	44
rotate_tor_circuit	45
run_direct_task	46

run_task	47
run_task_batch	51
search_options	54
summary.asa_agent	58
summary.asa_audit_result	59
summary.asa_benchmark	59
summary.asa_enumerate_result	60
summary.asa_evaluation_result	60
summary.asa_response	61
summary.asa_result	61
temporal_options	62
tor_options	63
write_csv.asa_enumerate_result	64

Index

65

as.data.frame.asa_audit_result	
Convert asa_audit_result to Data Frame	

Description

Convert asa_audit_result to Data Frame

Usage

```
## S3 method for class 'asa_audit_result'  
as.data.frame(x, ...)
```

Arguments

- x An asa_audit_result object
- ... Additional arguments (ignored)

Value

The audited data.frame with audit columns

as.data.frame.asa_enumerate_result	
Convert asa_enumerate_result to Data Frame	

Description

Convert asa_enumerate_result to Data Frame

Usage

```
## S3 method for class 'asa_enumerate_result'  
as.data.frame(x, ...)
```

Arguments

x	An asa_enumerate_result object
...	Additional arguments (ignored)

Value

The data data.frame from the result

```
as.data.frame.asa_evaluation_result
```

Convert asa_evaluation_result to Data Frame

Description

Convert asa_evaluation_result to Data Frame

Usage

```
## S3 method for class 'asa_evaluation_result'
as.data.frame(x, ...)
```

Arguments

x	An asa_evaluation_result object
...	Additional arguments (ignored)

Value

The per-task evaluation records as a data.frame

```
as.data.frame.asa_result
```

Convert asa_result to Data Frame

Description

Convert asa_result to Data Frame

Usage

```
## S3 method for class 'asa_result'
as.data.frame(x, ...)
```

Arguments

x	An asa_result object
...	Additional arguments (ignored)

Value

A single-row data frame

asa_agent	<i>Constructor for asa_agent Objects</i>
-----------	--

Description

Creates an S3 object representing an initialized ASA search agent.

Usage

```
asa_agent(python_agent, backend, model, config, llm = NULL, tools = NULL)
```

Arguments

- | | |
|--------------|--|
| python_agent | The underlying Python agent object |
| backend | LLM backend name (e.g., "openai", "groq") |
| model | Model identifier |
| config | Agent configuration list |
| llm | Optional LLM object used by LangGraph |
| tools | Optional list of tools associated with the agent |

Value

An object of class asa_agent

asa_audit	<i>Audit Enumeration Results for Completeness and Quality</i>
-----------	---

Description

Validates enumeration results for completeness, consistency, and data quality using either Claude Code (CLI) or a LangGraph-based audit pipeline.

Usage

```
asa_audit(  
  result,  
  query = NULL,  
  known_universe = NULL,  
  checks = c("completeness", "consistency", "gaps", "anomalies"),  
  backend = c("claude_code", "langgraph"),  
  claude_model = "claude-sonnet-4-20250514",  
  llm_model = "gpt-4.1-mini",  
  interactive = FALSE,  
  confidence_threshold = 0.8,  
  timeout = 120,  
  verbose = TRUE,  
  agent = NULL  
)
```

Arguments

result	An <code>asa_enumerate_result</code> object or a <code>data.frame</code> to audit
query	The original enumeration query (inferred from result if NULL)
known_universe	Optional vector of expected items for completeness check
checks	Character vector of checks to perform. Options: "completeness", "consistency", "gaps", "anomalies". Default runs all checks.
backend	Backend to use for auditing: "claude_code" (CLI) or "langgraph"
claude_model	Model to use with Claude Code backend
llm_model	Model to use with LangGraph backend
interactive	If TRUE and using <code>claude_code</code> backend, spawn an interactive Claude Code session instead of programmatic invocation
confidence_threshold	Flag items with confidence below this threshold
timeout	Timeout in seconds for the audit operation
verbose	Print progress messages
agent	Existing <code>asa_agent</code> for LangGraph backend (optional)

Details

The audit function adds three columns to the data:

- `_audit_flag`: "ok", "warning", or "suspect"
- `_audit_notes`: Explanation of any issues
- `_confidence_adjusted`: Revised confidence after audit

Audit Checks

completeness: Checks for missing items by comparing against `known_universe` (if provided) or using domain knowledge.

consistency: Validates data types, patterns, and value ranges.

gaps: Identifies systematic patterns of missing data (geographic, temporal, categorical gaps).

anomalies: Detects duplicates, outliers, and suspicious patterns.

Value

An `asa_audit_result` object containing:

data	Original data with audit columns added (<code>_audit_flag</code> , <code>_audit_notes</code>)
audit_summary	High-level summary of findings
issues	List of identified issues with severity and descriptions
recommendations	Suggested remediation queries
completeness_score	0-1 score for data completeness
consistency_score	0-1 score for data consistency

Examples

```
## Not run:
# Audit enumeration results with Claude Code
senators <- asa_enumerate(
  query = "Find all current US senators",
  schema = c(name = "character", state = "character", party = "character")
)
audit <- asa_audit(senators, backend = "claude_code")
print(audit)

# Audit with known universe for precise completeness check
audit <- asa_audit(senators, known_universe = state.abb)

# Interactive mode for complex audits
asa_audit(senators, backend = "claude_code", interactive = TRUE)

# Use LangGraph backend
audit <- asa_audit(senators, backend = "langgraph", agent = agent)

## End(Not run)
```

asa_audit_result	<i>Constructor for asa_audit_result Objects</i>
------------------	---

Description

Creates an S3 object representing the result of a data quality audit.

Usage

```
asa_audit_result(
  data,
  audit_summary,
  issues,
  recommendations,
  completeness_score,
  consistency_score,
  backend_used,
  elapsed_time,
  query = NULL,
  checks = NULL
)
```

Arguments

data	data.frame with original data plus audit columns (_audit_flag, _audit_notes)
audit_summary	Character string with high-level findings
issues	List of identified issues with severity and descriptions
recommendations	Character vector of suggested remediation queries

completeness_score	Numeric 0-1 score for data completeness
consistency_score	Numeric 0-1 score for data consistency
backend_used	Which backend performed the audit ("claude_code" or "langgraph")
elapsed_time	Execution time in seconds
query	The original query (if available)
checks	Character vector of checks that were performed

Value

An object of class `asa_audit_result`

asa_config	<i>Create ASA Configuration Object</i>
------------	--

Description

Creates a configuration object that encapsulates all settings for ASA tasks. This provides a unified way to configure backend, model, search, temporal, and resource settings in a single object.

Usage

```
asa_config(  
  agent_backend = NULL,  
  backend = NULL,  
  model = NULL,  
  conda_env = NULL,  
  proxy = NA,  
  use_browser = NULL,  
  workers = NULL,  
  timeout = NULL,  
  rate_limit = NULL,  
  memory_folding = NULL,  
  memory_threshold = NULL,  
  memory_keep_recent = NULL,  
  use_observational_memory = NULL,  
  om_observation_token_budget = NULL,  
  om_reflection_token_budget = NULL,  
  om_buffer_tokens = NULL,  
  om_buffer_activation = NULL,  
  om_block_after = NULL,  
  om_async_prebuffer = NULL,  
  om_cross_thread_memory = NULL,  
  temporal = NULL,  
  search = NULL,  
  tor = NULL,  
  recursion_limit = NULL  
)
```


Arguments

agent_backend	Agent runtime backend: "agent" (default ASA LangGraph pipeline), "free-code" (headless free-code backend with ASA-managed provider routing and search MCP tools), or "opencode" (OpenCode CLI backend with ASA-managed provider routing and search MCP tools).
backend	LLM backend: "openai", "groq", "xai", "gemini", "exo", "openrouter", "anthropic", or "bedrock"
model	Model identifier (e.g., "gpt-4.1-mini")
conda_env	Conda environment name. Defaults to the package option <code>asa.default_conda_env</code> (or "asa_env" if unset).
proxy	Proxy URL (e.g., Tor SOCKS5). Use NA (default) to auto-detect from environment variables (ASA_PROXY, HTTP_PROXY, HTTPS_PROXY); use NULL to disable proxying.
use_browser	Enable Selenium browser tier for DuckDuckGo search.
workers	Number of parallel workers for batch operations
timeout	Request timeout in seconds
rate_limit	Requests per second
memory_folding	Enable DeepAgent-style memory folding
memory_threshold	Messages before folding triggers
memory_keep_recent	Exchanges to preserve after folding. An exchange is a user turn plus the assistant response, including any tool calls and tool outputs.
use_observational_memory	Enable observational memory pipeline.
om_observation_token_budget	Token budget for stored observation messages.
om_reflection_token_budget	Token budget for reflection summaries.
om_buffer_tokens	Token budget for buffered context before reflection.
om_buffer_activation	Fraction of buffer usage that triggers reflection.
om_block_after	Fraction of context usage where buffering blocks generation.
om_async_prebuffer	Whether to pre-buffer observations asynchronously.
om_cross_thread_memory	Whether observational memory is shared across threads.
temporal	Temporal filtering options (use <code>temporal_options()</code>)
search	Search configuration (use <code>search_options()</code>)
tor	Tor registry options (use <code>tor_options()</code>)
recursion_limit	Optional default maximum number of agent steps. When NULL, mode-specific defaults are used at runtime unless overridden in run_task/run_task_batch .

Details

The configuration object can be passed to `run_task()`, `run_task_batch()`, `asa_enumerate()`, and other functions to provide consistent settings across operations.

Value

An object of class `asa_config`

See Also

[temporal_options](#), [search_options](#)

Examples

```
## Not run:
# Create configuration
config <- asa_config(
  backend = "openai",
  model = "gpt-4.1-mini",
  workers = 4,
  temporal = temporal_options(time_filter = "y")
)

# Use with run_task
result <- run_task("Find recent AI breakthroughs", config = config)

## End(Not run)
```

asa_enumerate

Multi-Agent Research for Open-Ended Queries

Description

Performs intelligent open-ended research tasks using multi-agent orchestration. Decomposes complex queries into sub-tasks, executes parallel searches, and aggregates results into structured output (data.frame, CSV, or JSON).

Usage

```
asa_enumerate(
  query = NULL,
  schema = NULL,
  output = c("data.frame", "csv", "json"),
  config = NULL,
  workers = NULL,
  max_rounds = NULL,
  budget = list(queries = 50L, tokens = 200000L, time_sec = 300L),
  stop_policy = list(target_items = NULL, plateau_rounds = 2L, novelty_min = 0.05,
    novelty_window = 20L),
  sources = list(web = TRUE, wikipedia = TRUE, wikidata = TRUE),
  allow_read_webpages = NULL,
```

```

webpage_relevance_mode = NULL,
webpage_embedding_provider = NULL,
webpage_embedding_model = NULL,
temporal = NULL,
pagination = TRUE,
progress = TRUE,
include_provenance = FALSE,
checkpoint = TRUE,
checkpoint_dir = tempdir(),
resume_from = NULL,
agent = NULL,
backend = NULL,
model = NULL,
conda_env = NULL,
verbose = TRUE
)

```

Arguments

query	Character string describing the research goal. Required for fresh runs; optional when resume_from points to an existing checkpoint. Examples: "Find all current US senators with their state, party, and term end date"
schema	Named character vector defining the output schema. Names are column names, values are R types ("character", "numeric", "logical"). Use NULL or "auto" for LLM-proposed schema.
output	Output format: "data.frame" (default), "csv", or "json".
config	Optional asa_config object to supply defaults for agent initialization and per-run search/temporal settings. Explicit arguments to asa_enumerate() take precedence over config.
workers	Number of parallel search workers. Defaults to value from config\$workers when provided, otherwise ASA_DEFAULT_WORKERS (typically 4).
max_rounds	Maximum research iterations. Defaults to value from ASA_DEFAULT_MAX_ROUNDS (typically 8).
budget	Named list with resource limits: <ul style="list-style-type: none"> queries: Maximum search queries (default: 50) tokens: Maximum LLM tokens (default: 200000) time_sec: Maximum execution time in seconds (default: 300)
stop_policy	Named list with stopping criteria: <ul style="list-style-type: none"> target_items: Stop when this many items found (NULL = unknown) plateau_rounds: Stop after N rounds with no new items (default: 2) novelty_min: Minimum new items ratio per round (default: 0.05) novelty_window: Window size for novelty calculation (default: 20)
sources	Named list controlling which sources to use: <ul style="list-style-type: none"> web: Use DuckDuckGo web search (default: TRUE) wikipedia: Use Wikipedia (default: TRUE) wikidata: Use Wikidata SPARQL for authoritative enumerations (default: TRUE)

allow_read_webpages	If TRUE, the agent may open and read full webpages (in addition to search snippets) when it helps extraction. Use NULL (default) to inherit from config\$search (when provided); otherwise defaults to FALSE for safety and to avoid large context usage.
webpage_relevance_mode	Relevance selection for opened webpages. One of: "auto" (default), "lexical", "embeddings". When "embeddings" or "auto" with an available provider, the tool uses vector similarity to pick the most relevant excerpts; otherwise it falls back to lexical overlap.
webpage_embedding_provider	Embedding provider to use for relevance. One of: "auto" (default), "openai", "sentence_transformers".
webpage_embedding_model	Embedding model identifier. For OpenAI, defaults to "text-embedding-3-small". For sentence-transformers, use a local model name (e.g., "all-MiniLM-L6-v2").
temporal	Named list for temporal filtering: <ul style="list-style-type: none"> • after: ISO 8601 date string (e.g., "2020-01-01") - results after this date • before: ISO 8601 date string (e.g., "2024-01-01") - results before this date • time_filter: DuckDuckGo time filter ("d", "w", "m", "y") for day/week/month/year • strictness: "best_effort" (default) or "strict" (verifies dates via metadata) • use_wayback: Use Wayback Machine for strict pre-date guarantees (default: FALSE)
pagination	Enable pagination for large result sets (default: TRUE).
progress	Show progress bar and status updates (default: TRUE).
include_provenance	Include source URLs and confidence per row (default: FALSE).
checkpoint	Enable auto-save after planning and each completed round (default: TRUE).
checkpoint_dir	Directory for checkpoint files (default: tempdir()).
resume_from	Path to checkpoint file to resume from (default: NULL). For v2 checkpoints this continues execution from the last saved round state. Legacy v1 checkpoints return the saved result without continuation.
agent	An initialized asa_agent object. If NULL, uses the current agent or creates a new one with specified backend/model.
backend	LLM backend if creating new agent: "openai", "groq", "xai", "gemini", "exo", "openrouter", "anthropic", or "bedrock".
model	Model identifier if creating new agent.
conda_env	Conda environment name. Defaults to the package option asa.default_conda_env (or "asa_env" if unset).
verbose	Print status messages (default: TRUE).

Details

The function uses an iterative graph architecture:

1. **Planner:** Decomposes the query and proposes a search plan.
2. **Searcher:** Queries Wikidata (when applicable) and falls back to web/Wikipedia.
3. **Deduper:** Removes duplicates using hashing + fuzzy matching.

4. **Stopper:** Evaluates stopping criteria (novelty, budgets, saturation).

Parallelism is limited and backend-dependent. For example, strict temporal filtering may verify publication dates in parallel up to workers.

For known entity types (US senators, countries, Fortune 500), Wikidata provides authoritative enumerations with complete, verified data.

Value

An object of class `asa_enumerate_result` containing:

- `data`: `data.frame` with results matching the schema
- `status`: "complete", "partial", or "failed"
- `stop_reason`: Why the search stopped
- `metrics`: List with rounds, queries_used, novelty_curve, coverage
- `provenance`: If `include_provenance=TRUE`, source info per row
- `checkpoint_file`: Path to checkpoint if saved

Checkpointing

With `checkpoint=TRUE`, state is saved after planning and after each completed round. If interrupted, use `resume_from` to continue from the last checkpoint:

```
result <- asa_enumerate(resume_from = "/path/to/checkpoint.rds")
```

Schema

The schema defines expected output columns:

```
schema = c(name = "character", state = "character", party = "character")
```

With `schema = "auto"`, the planner agent proposes a schema based on the query.

See Also

[run_task](#), [initialize_agent](#)

Examples

```
## Not run:
# Find all US senators
senators <- asa_enumerate(
  query = "Find all current US senators with state, party, and term end date",
  schema = c(name = "character", state = "character",
             party = "character", term_end = "character"),
  stop_policy = list(target_items = 100),
  include_provenance = TRUE
)
head(senators$data)

# Find countries with auto schema
countries <- asa_enumerate(
  query = "Find all countries with their capitals and populations",
  schema = "auto",
```

```

    output = "csv"
  )

  # Resume from checkpoint
  result <- asa_enumerate(
    query = "Find Fortune 500 CEOs",
    resume_from = "/tmp/asa_enumerate_abc123.rds"
  )

  # Temporal filtering: results from specific date range
  companies_2020s <- asa_enumerate(
    query = "Find tech companies founded recently",
    temporal = list(
      after = "2020-01-01",
      before = "2024-01-01",
      strictness = "best_effort"
    )
  )

  # Temporal filtering: past year with DuckDuckGo time filter
  recent_news <- asa_enumerate(
    query = "Find AI research breakthroughs",
    temporal = list(
      time_filter = "y" # past year
    )
  )

  # Strict temporal filtering with Wayback Machine
  historical <- asa_enumerate(
    query = "Find Fortune 500 companies",
    temporal = list(
      before = "2015-01-01",
      strictness = "strict",
      use_wayback = TRUE
    )
  )

  ## End(Not run)

```

asa_enumerate_result *Constructor for asa_enumerate_result Objects*

Description

Creates an S3 object representing the result of an enumeration task.

Usage

```

asa_enumerate_result(
  data,
  status,
  stop_reason,
  metrics,

```

```

    provenance = NULL,
    plan = NULL,
    checkpoint_file = NULL,
    query = NULL,
    schema = NULL
  )

```

Arguments

data	data.frame containing the enumeration results
status	Result status: "complete", "partial", or "failed"
stop_reason	Why the enumeration stopped (e.g., "target_reached", "novelty_plateau")
metrics	List with execution metrics (rounds, queries_used, etc.)
provenance	Optional data.frame with source information per row
plan	The enumeration plan from the planner agent
checkpoint_file	Path to saved checkpoint file
query	The original enumeration query
schema	The schema used for extraction

Value

An object of class `asa_enumerate_result`

`asa_evaluation_result` *Constructor for asa_evaluation_result Objects*

Description

Creates an S3 object representing the result of an agent evaluation run.

Usage

```

asa_evaluation_result(
  benchmark,
  task_results,
  run_summaries,
  metrics,
  agent_spec,
  config_snapshot,
  outputs = NULL,
  elapsed_time = NA_real_
)

```

Arguments

benchmark	Benchmark object used for the evaluation
task_results	data.frame with one row per task execution
run_summaries	data.frame with one row per evaluation run
metrics	Aggregate evaluation metrics
agent_spec	Named list describing the evaluated agent
config_snapshot	Configuration snapshot used during evaluation
outputs	Optional named list of retained raw outputs
elapsed_time	Total elapsed evaluation time in minutes

Value

An object of class `asa_evaluation_result`

<code>asa_response</code>	<i>Constructor for <code>asa_response</code> Objects</i>
---------------------------	--

Description

Creates an S3 object representing an agent response.

Usage

```
asa_response(
  message,
  status_code,
  raw_response,
  trace,
  elapsed_time,
  prompt,
  trace_json = "",
  fold_stats = list(),
  thread_id = NULL,
  stop_reason = NULL,
  budget_state = list(),
  field_status = list(),
  diagnostics = list(),
  json_repair = list(),
  final_payload = NULL,
  terminal_valid = FALSE,
  terminal_payload_hash = NULL,
  payload_integrity = list(),
  completion_gate = list(),
  tokens_used = NA_integer_,
  input_tokens = NA_integer_,
  output_tokens = NA_integer_,
  token_trace = list()
)
```


Arguments

message	The final response text
status_code	Status code (200 = success, 100 = error)
raw_response	The full Python response object
trace	Full text trace of agent execution
elapsed_time	Execution time in minutes
prompt	The original prompt
trace_json	Structured JSON trace (when available)
fold_stats	Diagnostic metrics from memory folding (list). Includes fold_count (integer or error message string), fold_messages_removed, fold_total_messages_removed, fold_chars_input, fold_summary_chars (effective per-fold summary growth chars), fold_summary_total_chars, fold_summary_delta_chars, fold_trigger_reason, fold_safe_boundary_idx, fold_compression_ratio, fold_parse_success, and fold_summarizer_latency_m.
thread_id	Resolved LangGraph thread identifier used for this run.
stop_reason	Terminal stop reason from agent state (when available).
budget_state	Tool-call budget state snapshot from agent state.
field_status	Per-field extraction status map from agent state.
diagnostics	Runtime diagnostics counters and intervention notes.
json_repair	JSON repair/fallback events emitted during execution.
final_payload	Canonical terminal payload from agent state (when available).
terminal_valid	Logical flag indicating whether a schema-valid terminal payload was emitted.
terminal_payload_hash	Canonical hash of terminal payload from agent state.
payload_integrity	Diagnostics about released payload provenance, repair, and truncation markers.
completion_gate	Deterministic outcome-verification report from agent state.
tokens_used	Total token count for this invocation (integer, or NA).
input_tokens	Input (prompt) token count (integer, or NA).
output_tokens	Output (completion) token count (integer, or NA).
token_trace	Per-step token usage trace (list).

Value

An object of class `asa_response`

asa_result

*Constructor for asa_result Objects***Description**

Creates an S3 object representing the result of a research task.

Usage

```
asa_result(
  prompt,
  message,
  parsed,
  raw_output,
  elapsed_time,
  status,
  search_tier = "unknown",
  parsing_status = NULL,
  trace_json = "",
  execution = NULL
)
```

Arguments

prompt	The original prompt
message	The agent's response text
parsed	Parsed output (list or NULL)
raw_output	Full agent trace
elapsed_time	Execution time in minutes
status	Status ("success" or "error")
search_tier	Which search tier was used ("primp", "selenium", "ddgs", "requests", or "unknown"). Useful for assessing result quality.
parsing_status	List with JSON parsing validation info: valid (logical), reason ("ok", "parsing_failed", "not_object", "missing_fields", "null_values", "no_validation"), and missing (character vector of missing/invalid fields).
trace_json	Structured JSON trace (when available)
execution	Optional operational metadata list. Common fields include thread_id, stop_reason, status_code, tool budget counters, fold_count, and action/plan metadata extracted from agent state. When created via run_task , selected execution fields are also attached as top-level aliases such as token_stats, fold_stats, plan, and action_ascii.

Value

An object of class `asa_result`

benchmark_from_df	Create an ASA Benchmark from a Data Frame
-------------------	---

Description

Converts a tabular benchmark definition into an asa_benchmark.

Usage

```
benchmark_from_df(
  df,
  prompt_col,
  expected_col,
  id_col = NULL,
  category_col = NULL,
  output_format_col = NULL,
  match_col = NULL,
  run_args_col = NULL,
  field_matchers_col = NULL,
  name = "CustomBenchmark",
  description = NULL,
  version = "1.0",
  citation = NULL,
  metadata = list(),
  output_format = "text",
  match = "exact"
)
```

Arguments

df	data.frame containing benchmark rows
prompt_col	Column containing prompts
expected_col	Column containing expected values
id_col	Optional column containing task ids
category_col	Optional column containing task categories
output_format_col	Optional column containing per-task output formats
match_col	Optional column containing per-task match modes
run_args_col	Optional column containing per-task run arguments
field_matchers_col	Optional column containing per-task field matcher specs
name	Benchmark name (default: "CustomBenchmark")
description	Optional benchmark description
version	Benchmark version string (default: "1.0")
citation	Optional citation string
metadata	Optional benchmark metadata
output_format	Default output format when output_format_col is NULL
match	Default matcher when match_col is NULL

Value

An object of class `asa_benchmark`

<code>benchmark_from_yaml</code>	<i>Load an ASA Benchmark from YAML</i>
----------------------------------	--

Description

Reads a benchmark definition from a YAML file.

Usage

`benchmark_from_yaml(path)`

Arguments

`path` Path to YAML benchmark file

Value

An object of class `asa_benchmark`

<code>build_backend</code>	<i>Build the Python Backend Environment</i>
----------------------------	---

Description

Creates a conda environment with all required Python dependencies for the asa search agent, including LangChain, LangGraph, and search tools.

Usage

```
build_backend(  
    conda_env = NULL,  
    conda = "auto",  
    python_version = "3.12",  
    force = FALSE,  
    check_browser = TRUE,  
    fix_browser = FALSE,  
    chromedriver_dir = NULL  
)
```

Arguments

conda_env	Name of the conda environment. Defaults to the package option <code>asa.default_conda_env</code> (or <code>"asa_env"</code> if unset).
conda	Path to conda executable (default: <code>"auto"</code>)
python_version	Python version to use (default: <code>"3.12"</code>)
force	If TRUE, delete and recreate the conda environment if it already exists.
check_browser	If TRUE, performs a best-effort check for a system Chrome/Chromium binary and chromedriver, warning when major versions are incompatible. Set to FALSE to skip this check.
fix_browser	If TRUE and <code>check_browser = TRUE</code> , attempt to download a matching chromedriver for the detected Chrome version and prepend it to PATH for the current R session. The driver is stored under <code>chromedriver_dir</code> .
chromedriver_dir	Optional directory to store downloaded chromedriver binaries. Defaults to <code>~/ .asa/chromedriver</code> when <code>fix_browser = TRUE</code> .

Details

This function creates a new conda environment and installs the following Python packages:

- langchain_groq, langchain_community, langchain_openai, langchain_google_genai
- langgraph
- ddgs (DuckDuckGo search)
- selenium, primp (browser automation)
- undetected-chromedriver (stealth Chrome)
- beautifulsoup4, curl_cffi, requests
- langextract (schema extraction from webpage text)
- fake_headers, httpx
- stem (Tor control)
- pysocks, socksio (proxy support)

Python compatibility note: Current LangChain releases still import parts of `pydantic.v1`, which emits warnings on Python 3.14+. Prefer Python 3.12 (or 3.13) until upstream dependencies remove that compatibility layer.

Value

Invisibly returns NULL; called for side effects.

Examples

```
## Not run:
# Create the default environment
build_backend()

# Rebuild from scratch
build_backend(force = TRUE)

# Create with a custom name
build_backend(conda_env = "my_asa_env")
```

```
## End(Not run)
```

build_prompt	<i>Build a Task Prompt from Template</i>
--------------	--

Description

Creates a formatted prompt by substituting variables into a template.

Usage

```
build_prompt(template, ...)
```

Arguments

template	A character string with placeholders in the form {variable_name}
...	Named arguments to substitute into the template

Value

A formatted prompt string

Examples

```
## Not run:
prompt <- build_prompt(
  template = "Find information about {{name}} in {{country}} during {{year}}",
  name = "Marie Curie",
  country = "France",
  year = 1903
)

## End(Not run)
```

check_backend	<i>Check Python Backend Availability</i>
---------------	--

Description

Checks whether the configured Conda environment exists and can import the Python modules required by the ASA backend, without initializing reticulate's Python session for the current R process.

Usage

```
check_backend(conda_env = NULL, conda = "auto", strict = FALSE)
```

Arguments

conda_env	Name of the Conda environment. Defaults to the package option <code>asa.default_conda_env</code> (or <code>"asa_env"</code> if unset).
conda	Path to conda executable (default: <code>"auto"</code>)
strict	If TRUE, also require optional packages used for Tor control and stealth Chrome support (e.g., <code>stem</code> , <code>undetected_chromedriver</code>).

Value

A list with components:

- `available`: Logical, TRUE if environment is ready
- `conda_env`: Name of the environment checked
- `python_version`: Python version if available
- `missing_packages`: Character vector of missing required packages (if any)
- `missing_optional_packages`: Character vector of missing optional packages (if any)
- `strict`: Logical, whether optional packages were required

Examples

```
## Not run:
status <- check_backend()
if (!status$available) {
  build_backend()
}

## End(Not run)
```

configure_search

Configure Python Search Parameters

Description

Sets global configuration values for the Python search module. These values control timeouts, retry behavior, and result limits.

Usage

```
configure_search(
  max_results = NULL,
  timeout = NULL,
  max_retries = NULL,
  retry_delay = NULL,
  backoff_multiplier = NULL,
  captcha_backoff_base = NULL,
  page_load_wait = NULL,
  inter_search_delay = NULL,
  conda_env = NULL,
  selenium_browser_preference = NULL
)
```

Arguments

max_results	Maximum number of search results to return (default: 10)
timeout	HTTP request timeout in seconds (default: 30)
max_retries	Maximum retry attempts on failure (default: 3)
retry_delay	Initial delay between retries in seconds (default: 2)
backoff_multiplier	Multiplier for exponential backoff (default: 1.5)
captcha_backoff_base	Base multiplier for CAPTCHA backoff (default: 3)
page_load_wait	Wait time after page load in seconds (default: 2)
inter_search_delay	Delay between consecutive searches in seconds (default: 1.5)
conda_env	Name of the conda environment. Defaults to the package option <code>asa.default_conda_env</code> (or <code>"asa_env"</code> if unset).
selenium_browser_preference	Selenium browser engine preference order. One of <code>"firefox_first"</code> (default) or <code>"chrome_first"</code> .

Value

Invisibly returns a list with the current configuration

Examples

```
## Not run:
# Increase timeout for slow connections
configure_search(timeout = 30, max_retries = 5)

# Get more results
configure_search(max_results = 20)

# Add delay between searches to avoid rate limiting
configure_search(inter_search_delay = 2.0)

## End(Not run)
```

configure_search_logging

Configure Python Search Logging Level

Description

Sets the logging level for the Python search module. This controls how much diagnostic output is produced during web searches.

Usage

```
configure_search_logging(level = "WARNING", conda_env = NULL)
```


Arguments

level	Log level: "DEBUG", "INFO", "WARNING" (default), "ERROR", or "CRITICAL"
conda_env	Name of the conda environment. Defaults to the package option <code>asa.default_conda_env</code> (or "asa_env" if unset).

Details

Log levels from most to least verbose:

- **DEBUG:** Detailed diagnostic information for debugging
- **INFO:** General operational information
- **WARNING:** Indicates something unexpected but not an error (default)
- **ERROR:** Serious problems that prevented an operation
- **CRITICAL:** Very serious errors

Value

Invisibly returns the current logging level

Examples

```
## Not run:
# Enable verbose debugging output
configure_search_logging("DEBUG")

# Run a search (will show detailed logs)
result <- run_task("What is the population of Tokyo?", agent = agent)

# Disable verbose output
configure_search_logging("WARNING")

## End(Not run)
```

configure_temporal	<i>Configure Temporal Filtering for Search</i>
--------------------	--

Description

Sets or clears temporal filtering on the DuckDuckGo search tool. This affects all subsequent searches until changed or cleared.

Usage

```
configure_temporal(time_filter = NULL)
```

Arguments

time_filter	DuckDuckGo time filter: "d" (day), "w" (week), "m" (month), "y" (year), or NULL/NA/"none" to clear
-------------	--

Details

This function modifies the search tool's time parameter, which is passed to DuckDuckGo as the df parameter. The filter restricts results to content indexed within the specified time period.

Note: This only affects DuckDuckGo searches. For Wikidata queries with temporal filtering, use `asa_enumerate()` with its temporal parameter.

Value

Invisibly returns the previous time filter setting

Time Filter Values

- "d": Past 24 hours (day)
- "w": Past 7 days (week)
- "m": Past 30 days (month)
- "y": Past 365 days (year)
- NULL, NA, or "none": No time restriction (default)

See Also

[run_task](#), [asa_enumerate](#)

Examples

```
## Not run:
# Restrict to past year
configure_temporal("y")
result <- run_task("Find recent AI breakthroughs", agent = agent)

# Clear temporal filter
configure_temporal(NULL)

# Past week only
configure_temporal("w")

## End(Not run)
```

configure_tor_registry

Configure Tor Exit Registry

Description

Sets up the shared Tor exit health registry used by the Python search stack to avoid reusing tainted or overused exit nodes.

Usage

```

configure_tor_registry(
    registry_path = NULL,
    enable = ASA_TOR_REGISTRY_ENABLED,
    bad_ttl = ASA_TOR_BAD_TTL,
    good_ttl = ASA_TOR_GOOD_TTL,
    overuse_threshold = ASA_TOR_OVERUSE_THRESHOLD,
    overuse_decay = ASA_TOR_OVERUSE_DECAY,
    max_rotation_attempts = ASA_TOR_MAX_ROTATION_ATTEMPTS,
    ip_cache_ttl = ASA_TOR_IP_CACHE_TTL,
    conda_env = NULL
)

```

Arguments

registry_path	Path to the SQLite registry file (default: user cache).
enable	Enable the registry (set FALSE to disable tracking).
bad_ttl	Seconds to keep a bad/tainted exit before reuse.
good_ttl	Seconds to treat an exit as good before refreshing.
overuse_threshold	Maximum recent uses before a good exit is treated as overloaded.
overuse_decay	Window (seconds) for counting recent uses before decay.
max_rotation_attempts	Maximum rotations to find a clean exit.
ip_cache_ttl	Seconds to cache exit IP lookups.
conda_env	Conda environment name for the Python module. Defaults to the package option <code>asa.default_conda_env</code> (or "asa_env" if unset).

Value

Invisibly returns a list of the configured values (or NULL on error).

create_benchmark	<i>Create an ASA Benchmark Object</i>
------------------	---------------------------------------

Description

Creates a structured benchmark specification for agent evaluation.

Usage

```

create_benchmark(
    name,
    tasks,
    description = NULL,
    version = "1.0",
    citation = NULL,
    metadata = list()
)

```

Arguments

name	Benchmark name
tasks	List of benchmark task definitions
description	Optional benchmark description
version	Benchmark version string (default: "1.0")
citation	Optional citation string
metadata	Optional named list of additional benchmark metadata

Value

An object of class `asa_benchmark`

<code>evaluate_agent</code>	<i>Evaluate an ASA Agent Against a Benchmark</i>
-----------------------------	--

Description

Runs a benchmark suite against an ASA agent using deterministic local scoring.

Usage

```
evaluate_agent(
  benchmark,
  agent = NULL,
  config = NULL,
  runs = 1L,
  fail_fast = FALSE,
  keep_outputs = FALSE,
  verbose = TRUE
)
```

Arguments

benchmark	An <code>asa_benchmark</code> object or built-in benchmark name
agent	Optional initialized <code>asa_agent</code>
config	Optional <code>asa_config</code> used to initialize an agent when agent is not supplied
runs	Number of repeated benchmark runs (default: 1)
fail_fast	If TRUE, stop evaluation on first execution failure
keep_outputs	If TRUE, retain raw <code>asa_result</code> objects in the returned evaluation result
verbose	Print progress messages

Value

An object of class `asa_evaluation_result`

extract_agent_results *Extract Structured Data from Agent Traces*

Description

Parses raw agent output to extract search snippets, Wikipedia content, URLs, JSON data, and search tier information. This is the main function for post-processing agent traces.

Usage

```
extract_agent_results(raw_output)
```

Arguments

raw_output Raw trace string from `asa_result$raw_output`, or a structured JSON trace (`asa_trace_v1`) from `asa_result$trace_json`

Value

A list with components:

- `search_snippets`: Character vector of search result content
- `search_urls`: Character vector of URLs from search results
- `wikipedia_snippets`: Character vector of Wikipedia content
- `json_data`: Extracted JSON data (canonical by default; see option `asa.enable_trace_field_mining`)
- `json_data_canonical`: Canonical terminal JSON extracted from trace
- `json_data_inferred`: Optional inferred JSON with trace-mining enabled
- `search_tiers`: Character vector of unique search tiers used (e.g., "primp", "selenium", "ddgs", "requests")

Examples

```
## Not run:
result <- run_task("Who is the president of France?", agent = agent)
trace <- if (!is.null(result$trace_json) && nzchar(result$trace_json)) {
  result$trace_json
} else {
  result$raw_output
}
extracted <- extract_agent_results(trace)
print(extracted$search_snippets)
print(extracted$search_tiers) # Shows which search tier was used

## End(Not run)
```

extract_search_snippets

Extract Search Snippets by Source Number

Description

Extracts content from Search tool messages in the agent trace.

Usage

```
extract_search_snippets(text, source = NULL)
```

Arguments

text	Structured JSON trace (asa_trace_v1)
source	Optional integer vector of source numbers to return (1-indexed). If NULL (default), returns a character vector for all sources encountered (with empty strings for missing indices).

Value

Character vector of search snippets, ordered by source number

Examples

```
## Not run:
snippets <- extract_search_snippets(response$trace)

## End(Not run)
```

extract_search_tiers *Extract Search Tier Information*

Description

Extracts which search tier was used from the agent trace. The search module uses a multi-tier fallback system:

- primp: Fast HTTP client with browser impersonation (Tier 0)
- selenium: Headless browser for JS-rendered content (Tier 1)
- ddgs: Standard DDGS Python library (Tier 2)
- requests: Raw POST to DuckDuckGo HTML endpoint (Tier 3)

Usage

```
extract_search_tiers(text)
```

Arguments

text Structured JSON trace (asa_trace_v1)

Value

Character vector of unique tier names encountered (e.g., "primp", "selenium", "ddgs", "requests")

Examples

```
## Not run:
tiers <- extract_search_tiers(response$trace)
print(tiers) # e.g., "primp"

## End(Not run)
```

extract_urls	<i>Extract URLs by Source Number</i>
--------------	--------------------------------------

Description

Extracts URLs from Search tool messages in the agent trace.

Usage

```
extract_urls(text, source = NULL)
```

Arguments

text Structured JSON trace (asa_trace_v1)

source Optional integer vector of source numbers to return (1-indexed). If NULL (default), returns a character vector for all sources encountered (with empty strings for missing indices).

Value

Character vector of URLs, ordered by source number

Examples

```
## Not run:
urls <- extract_urls(response$trace)

## End(Not run)
```

extract_wikipedia_content	<i>Extract Wikipedia Content</i>
---------------------------	----------------------------------

Description

Extracts content from Wikipedia tool messages in the agent trace.

Usage

```
extract_wikipedia_content(text)
```

Arguments

text	Structured JSON trace (asa_trace_v1)
------	--------------------------------------

Value

Character vector of Wikipedia snippets

Examples

```
## Not run:
wiki <- extract_wikipedia_content(response$trace)

## End(Not run)
```

get_agent	<i>Get the Current Agent</i>
-----------	------------------------------

Description

Returns the currently initialized agent, or NULL if not initialized.

Usage

```
get_agent()
```

Value

An asa_agent object or NULL

Examples

```
## Not run:
agent <- get_agent()
if (is.null(agent)) {
  agent <- initialize_agent()
}

## End(Not run)
```

get_tor_ip	<i>Get External IP via Tor</i>
------------	--------------------------------

Description

Retrieves the external IP address as seen through Tor proxy.

Usage

```
get_tor_ip(proxy = "socks5h://127.0.0.1:9050", timeout = 30L)
```

Arguments

proxy	Tor proxy URL (e.g., "socks5h://127.0.0.1:9050" for default, or "socks5h://127.0.0.1:9055" for instance on port 9055)
timeout	Timeout in seconds (default: 30). Useful for parallel workloads where some Tor exits may be slow.

Value

IP address string or NA on failure

Examples

```
## Not run:
# Default Tor instance
ip <- get_tor_ip()
message("Current Tor IP: ", ip)

# Check specific Tor instance (e.g., for parallel jobs)
ip <- get_tor_ip(proxy = "socks5h://127.0.0.1:9055")

## End(Not run)
```

initialize_agent	<i>Initialize the ASA Search Agent</i>
------------------	--

Description

Initializes the Python environment and creates the LangGraph agent with search tools (Wikipedia, DuckDuckGo). The agent can use multiple LLM backends and supports DeepAgent-style memory folding.

Usage

```
initialize_agent(
    agent_backend = NULL,
    backend = NULL,
    model = NULL,
    conda_env = NULL,
    proxy = NA,
    use_browser = ASA_DEFAULT_USE_BROWSER,
    search = NULL,
    use_memory_folding = ASA_DEFAULT_MEMORY_FOLDING,
    memory_threshold = ASA_DEFAULT_MEMORY_THRESHOLD,
    memory_keep_recent = ASA_DEFAULT_MEMORY_KEEP_RECENT,
    fold_char_budget = ASA_DEFAULT_FOLD_CHAR_BUDGET,
    use_observational_memory = ASA_DEFAULT_USE_OBSERVATIONAL_MEMORY,
    om_observation_token_budget = ASA_DEFAULT_OM_OBSERVATION_TOKENS,
    om_reflection_token_budget = ASA_DEFAULT_OM_REFLECTION_TOKENS,
    om_buffer_tokens = ASA_DEFAULT_OM_BUFFER_TOKENS,
    om_buffer_activation = ASA_DEFAULT_OM_BUFFER_ACTIVATION,
    om_block_after = ASA_DEFAULT_OM_BLOCK_AFTER,
    om_async_prebuffer = ASA_DEFAULT_OM_ASYNC_PREBUFFER,
    om_cross_thread_memory = ASA_DEFAULT_OM_CROSS_THREAD_MEMORY,
    rate_limit = ASA_DEFAULT_RATE_LIMIT,
    timeout = ASA_DEFAULT_TIMEOUT,
    tor = tor_options(),
    verbose = TRUE,
    recursion_limit = NULL
)
```

Arguments

agent_backend	Agent runtime backend. Use "agent" (default) for the built-in ASA Lang-Graph pipeline, "free-code" to run the external free-code agent behind ASA-managed provider and search wrappers, or "opencode" to run the OpenCode CLI behind the same ASA-managed provider and search wrappers.
backend	LLM backend to use. One of: "openai", "groq", "xai", "gemini", "exo", "open-router", "anthropic", "bedrock"
model	Model identifier (e.g., "gpt-4.1-mini", "llama-3.3-70b-versatile")
conda_env	Name of the conda environment with Python dependencies
proxy	Proxy URL for search tools. Use NA (default) to auto-detect from environment variables (ASA_PROXY, HTTP_PROXY, HTTPS_PROXY). Use NULL to disable proxying. Use ASA_DEFAULT_PROXY (or "socks5h://127.0.0.1:9050") to route searches through Tor.
use_browser	Enable Selenium browser tier for DuckDuckGo search (default: TRUE). Set to FALSE to avoid Chromedriver requirements.
search	Optional search options (created with search_options) used as defaults for run_task/asa_enumerate when no asa_config is provided. This is useful for setting OpenWebpage tuning knobs (e.g., webpage_max_chars, webpage_max_chunks, webpage_chunk_chars) once when initializing an agent.
use_memory_folding	Enable DeepAgent-style memory compression (default: TRUE)

memory_threshold	Number of messages before folding triggers (default: 10; message-count back-stop alongside fold_char_budget)
memory_keep_recent	Number of recent exchanges to preserve after folding (default: 4). An exchange is a user turn plus the assistant response, including any tool calls and tool outputs.
fold_char_budget	Total character budget across all messages before memory folding triggers (default: 30000). Lower values fold more aggressively; higher values preserve more raw conversation.
use_observational_memory	Enable observational memory pipeline.
om_observation_token_budget	Token budget for stored observation messages.
om_reflection_token_budget	Token budget for reflection summaries.
om_buffer_tokens	Token budget for buffered context before reflection.
om_buffer_activation	Fraction of buffer usage that triggers reflection.
om_block_after	Fraction of context usage where buffering blocks generation.
om_async_prebuffer	Whether to pre-buffer observations asynchronously.
om_cross_thread_memory	Whether observational memory is shared across threads.
rate_limit	Requests per second for rate limiting (default: 0.1)
timeout	Request timeout in seconds (default: 120)
tor	Tor registry options from tor_options . Disable shared tracking by setting <code>dirty_tor_exists = False</code> .
verbose	Print status messages (default: True)
recursion_limit	Optional default maximum number of agent steps to store in the initialized agent. When NULL (default), step limits fall back to mode-specific defaults at run time (memory folding: 100; standard agent: 20). Per-call <code>run_task(..., recursion_limit = ...)</code> overrides this value.

Details

The agent is created with these tools:

- Wikipedia: For looking up encyclopedic information
- DuckDuckGo Search: For web searches with a 4-tier fallback system (PRIMP -> Selenium -> DDGS library -> raw requests)
- OpenWebpage: (Optional) Fetch and extract readable text from a public webpage URL. This capability is disabled by default and must be enabled per-call (e.g., via `run_task(..., allow_read_webpages = True)`).

Memory folding (enabled by default) compresses older messages into a summary to manage context length in long conversations, following the DeepAgent paper.

Value

An object of class `asa_agent` containing the initialized agent and configuration.

API Keys

The following environment variables should be set based on your backend:

- OpenAI: `OPENAI_API_KEY`
- Groq: `GROQ_API_KEY`
- xAI: `XAI_API_KEY`
- Gemini: `GOOGLE_API_KEY` (or `GEMINI_API_KEY`)
- Anthropic: `ANTHROPIC_API_KEY`
- Bedrock: `AWS_ACCESS_KEY_ID` + `AWS_SECRET_ACCESS_KEY` (optionally `AWS_DEFAULT_REGION`, defaults to "us-east-1")
- OpenRouter: `OPENROUTER_API_KEY`
- Exo: no API key required (local server)

OpenRouter Models

When using the "openrouter" backend, model names must be in provider/model-name format. Examples:

- "openai/gpt-4o"
- "anthropic/claude-3-sonnet"
- "google/gemma-2-9b-it:free"
- "meta-llama/llama-3-70b-instruct"

See <https://openrouter.ai/models> for available models.

See Also

[run_task](#), [run_task_batch](#)

Examples

```
## Not run:
# Initialize with OpenAI
agent <- initialize_agent(
  backend = "openai",
  model = "gpt-4.1-mini"
)

# Initialize with Groq and custom settings
agent <- initialize_agent(
  backend = "groq",
  model = "llama-3.3-70b-versatile",
  use_memory_folding = FALSE,
  proxy = NULL # No Tor proxy
)

# Initialize with OpenRouter (access to 100+ models)
agent <- initialize_agent(
```

```
        backend = "openrouter",
        model = "anthropic/claude-3-sonnet" # Note: provider/model format
    )

    ## End(Not run)
```

is_tor_running	<i>Check if Tor is Running</i>
----------------	--------------------------------

Description

Checks if Tor is running and accessible on the default port.

Usage

```
is_tor_running(port = 9050L)
```

Arguments

port Port number (default: 9050)

Value

Logical indicating if Tor appears to be running

Examples

```
## Not run:
if (!is_tor_running()) {
  message("Start Tor with: brew services start tor")
}

## End(Not run)
```

list_benchmarks	<i>List Built-In ASA Benchmarks</i>
-----------------	-------------------------------------

Description

Lists benchmark definitions bundled with the package.

Usage

```
list_benchmarks()
```

Value

data.frame of benchmark metadata

load_benchmark	<i>Load a Built-In ASA Benchmark</i>
----------------	--------------------------------------

Description

Loads a benchmark bundled under inst/benchmarks/.

Usage

```
load_benchmark(name)
```

Arguments

name	Benchmark name
------	----------------

Value

An object of class asa_benchmark

print.asa_agent	<i>Print Method for asa_agent Objects</i>
-----------------	---

Description

Print Method for asa_agent Objects

Usage

```
## S3 method for class 'asa_agent'  
print(x, ...)
```

Arguments

x	An asa_agent object
...	Additional arguments (ignored)

Value

Invisibly returns the object

```
print.asa_audit_result
```

Print Method for asa_audit_result Objects

Description

Print Method for asa_audit_result Objects

Usage

```
## S3 method for class 'asa_audit_result'  
print(x, n = 6, ...)
```

Arguments

x	An asa_audit_result object
n	Number of data rows to preview (default: 6)
...	Additional arguments (ignored)

Value

Invisibly returns the object

```
print.asa_benchmark
```

Print Method for asa_benchmark Objects

Description

Print Method for asa_benchmark Objects

Usage

```
## S3 method for class 'asa_benchmark'  
print(x, ...)
```

Arguments

x	An asa_benchmark object
...	Additional arguments (ignored)

Value

Invisibly returns the benchmark object

print.asa_config	<i>Print Method for asa_config Objects</i>
------------------	--

Description

Print Method for asa_config Objects

Usage

```
## S3 method for class 'asa_config'  
print(x, ...)
```

Arguments

x	An asa_config object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.asa_enumerate_result	<i>Print Method for asa_enumerate_result Objects</i>
----------------------------	--

Description

Print Method for asa_enumerate_result Objects

Usage

```
## S3 method for class 'asa_enumerate_result'  
print(x, n = 6, ...)
```

Arguments

x	An asa_enumerate_result object
n	Number of data rows to preview (default: 6)
...	Additional arguments (ignored)

Value

Invisibly returns the object

```
print.asa_evaluation_result
```

Print Method for asa_evaluation_result Objects

Description

Print Method for asa_evaluation_result Objects

Usage

```
## S3 method for class 'asa_evaluation_result'  
print(x, ...)
```

Arguments

x	An asa_evaluation_result object
...	Additional arguments (ignored)

Value

Invisibly returns the evaluation result

```
print.asa_response
```

Print Method for asa_response Objects

Description

Print Method for asa_response Objects

Usage

```
## S3 method for class 'asa_response'  
print(x, ...)
```

Arguments

x	An asa_response object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.asa_result	<i>Print Method for asa_result Objects</i>
------------------	--

Description

Print Method for asa_result Objects

Usage

```
## S3 method for class 'asa_result'  
print(x, ...)
```

Arguments

x	An asa_result object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.asa_search	<i>Print Method for asa_search Objects</i>
------------------	--

Description

Print Method for asa_search Objects

Usage

```
## S3 method for class 'asa_search'  
print(x, ...)
```

Arguments

x	An asa_search object
...	Additional arguments (ignored)

print.asa_temporal	<i>Print Method for asa_temporal Objects</i>
--------------------	--

Description

Print Method for asa_temporal Objects

Usage

```
## S3 method for class 'asa_temporal'  
print(x, ...)
```

Arguments

x	An asa_temporal object
...	Additional arguments (ignored)

Value

Invisibly returns the object

print.asa_tor	<i>Print Method for asa_tor Objects</i>
---------------	---

Description

Print Method for asa_tor Objects

Usage

```
## S3 method for class 'asa_tor'  
print(x, ...)
```

Arguments

x	An asa_tor object
...	Additional arguments (ignored)

Value

Invisibly returns the object

process_outputs	<i>Process Multiple Agent Outputs</i>
-----------------	---------------------------------------

Description

Processes a data frame of raw agent outputs, extracting structured data.

Usage

```
process_outputs(df, parallel = FALSE, workers = 10L)
```

Arguments

df	Data frame with a 'raw_output' column containing agent traces
parallel	Use parallel processing
workers	Number of workers

Value

The input data frame with additional extracted columns: search_count, wiki_count, and any JSON fields found

reset_agent	<i>Reset the Agent</i>
-------------	------------------------

Description

Clears the initialized agent state, forcing reinitialization on next use. Also closes any open HTTP clients to prevent resource leaks.

Usage

```
reset_agent()
```

Value

Invisibly returns NULL

rotate_tor_circuit	<i>Rotate Tor Circuit (R-side, daemon restart)</i>
--------------------	--

Description

Requests a new Tor circuit by restarting the Tor service or sending SIGHUP.

Usage

```
rotate_tor_circuit(
    method = c("signal", "brew", "systemctl"),
    wait = 12L,
    pid = NULL
)
```

Arguments

method	Method to restart: "brew" (macOS), "systemctl" (Linux), or "signal"
wait	Seconds to wait for new circuit (default: 12)
pid	Optional PID of specific Tor process (only used with method="signal"). If NULL (default), finds the Tor process via pgrep.

Details

MEDIUM FIX: This function restarts the entire Tor daemon, which kills ALL circuits and affects parallel execution. For production use, prefer the Python-side control port rotation which sends SIGNAL NEWNYM to get a new circuit without restarting the daemon.

For parallel Tor setups with multiple instances, consider using Tor's built-in circuit rotation via MaxCircuitDirtiness and NewCircuitPeriod config options instead of this function.

Value

Invisibly returns TRUE on success, FALSE on failure

Note

The "brew" and "systemctl" methods restart the entire Tor daemon and should only be used as a last resort for recovery. The "signal" method is preferred but still affects all circuits on the process.

Examples

```
## Not run:
# Preferred: Use Python-side control port rotation (via run_task/asa_enumerate)
# This R function is for manual recovery only

# Send SIGHUP to Tor process (least disruptive)
rotate_tor_circuit(method = "signal")

# macOS with Homebrew (restarts daemon - use sparingly)
rotate_tor_circuit(method = "brew")

# Linux with systemd (restarts daemon - use sparingly)
```

```
rotate_tor_circuit(method = "systemctl")

## End(Not run)
```

run_direct_task	<i>Run a Direct Provider Task Without ASA Tooling</i>
-----------------	---

Description

Invokes the configured provider directly using the underlying chat model attached to an initialized agent. This bypasses the ASA agent loop, tools, search, and temporal prompt augmentation while preserving the same backend and model selection.

Usage

```
run_direct_task(
  prompt,
  output_format = "text",
  config = NULL,
  agent = NULL,
  verbose = FALSE
)
```

Arguments

prompt	The prompt or question to send directly to the configured LLM
output_format	Expected output format. One of: <ul style="list-style-type: none">• "text": Returns response text (default)• "json": Parse response as JSON• "raw": Include the raw provider response object for debugging• Character vector: Extract specific fields from response
config	An asa_config object for unified configuration, or NULL to use the currently initialized agent.
agent	An asa_agent object from initialize_agent , or NULL to initialize/reuse one from config.
verbose	Print progress messages (default: FALSE)

Value

An asa_result object with execution\$mode = "provider_direct" and search_tier = "none".

`run_task`*Run a Structured Task with the Agent*

Description

Executes a research task using the AI search agent with a structured prompt and returns parsed results. This is the primary function for running agent tasks.

Usage

```
run_task(
    prompt,
    output_format = "text",
    temporal = NULL,
    config = NULL,
    agent = NULL,
    expected_fields = NULL,
    expected_schema = NULL,
    thread_id = NULL,
    field_status = NULL,
    budget_state = NULL,
    search_budget_limit = NULL,
    unknown_after_searches = NULL,
    finalize_on_all_fields_resolved = NULL,
    field_rules = NULL,
    source_policy = NULL,
    retry_policy = NULL,
    finalization_policy = NULL,
    orchestration_options = NULL,
    performance_profile = NULL,
    webpage_policy = NULL,
    query_templates = NULL,
    verbose = FALSE,
    allow_read_webpages = NULL,
    webpage_relevance_mode = NULL,
    webpage_embedding_provider = NULL,
    webpage_embedding_model = NULL,
    use_plan_mode = FALSE,
    recursion_limit = NULL
)
```

Arguments

<code>prompt</code>	The task prompt or question for the agent to research
<code>output_format</code>	Expected output format. One of: <ul style="list-style-type: none">"text": Returns response text (default)"json": Parse response as JSON"raw": Include the raw Python response object for low-level debugging in addition to the normal trace fieldsCharacter vector: Extract specific fields from response

temporal	<p>Named list or <code>asa_temporal</code> object for temporal filtering:</p> <ul style="list-style-type: none"> • <code>time_filter</code>: DuckDuckGo time filter - "d" (day), "w" (week), "m" (month), "y" (year) • <code>after</code>: ISO 8601 date (e.g., "2020-01-01") - hint for results after this date (added to prompt context) • <code>before</code>: ISO 8601 date (e.g., "2024-01-01") - hint for results before this date (added to prompt context)
config	An <code>asa_config</code> object for unified configuration, or NULL to use defaults
agent	An <code>asa_agent</code> object from <code>initialize_agent</code> , or NULL to use the currently initialized agent
expected_fields	Optional character vector of field names expected in JSON output. When provided, validates that all fields are present and non-null. The result will include a <code>parsing_status</code> field with validation details.
expected_schema	Optional JSON schema tree used for best-effort repair when the agent output is missing required keys (especially when <code>recursion_limit</code> is reached). This should be a nested list describing the required JSON object/array shape, following the conventions used in <code>asa/tests/testthat/test-langgraph-remainingsteps.R</code> . When provided, this bypasses prompt-based schema inference. Structured terminal payloads automatically include sibling metadata keys <code><field>_source</code> (URL or NULL) and <code><field>_confidence</code> (numeric score in <code>[0,1]</code> or NULL) for top-level schema fields, including unresolved fields where sibling values are emitted as NULL, even when those sibling keys are not listed in <code>expected_schema</code> .
thread_id	Optional stable identifier for memory folding sessions. When provided, the same thread ID is reused so folded summaries persist across invocations. Defaults to NULL (new thread each call).
field_status	Optional per-field extraction ledger seed passed into the LangGraph state. Useful for resuming partially resolved schemas.
budget_state	Optional tool budget state seed passed into the LangGraph state (e.g., to resume prior progress).
search_budget_limit	Optional integer maximum number of Search tool calls for this run.
unknown_after_searches	Optional integer threshold after which unresolved fields may be marked as unknown.
finalize_on_all_fields_resolved	Optional logical flag. When TRUE, the agent finalizes once all required fields are resolved.
field_rules	Optional named list/dict of field-level extraction rules passed directly to the backend state. Use task-agnostic rule objects; NULL leaves backend defaults unchanged.
source_policy	Optional named list/dict controlling source selection and evidence-policy behavior in the backend state.
retry_policy	Optional named list/dict controlling backend retry behavior for follow-up planning and field resolution.
finalization_policy	Optional named list/dict controlling terminal finalization behavior in the backend state.

orchestration_options	Optional named list/dict of generic orchestration controls (component enable/mode and thresholds). This is passed directly to the backend state and must remain task-agnostic.
performance_profile	Optional orchestration profile. One of: "latency", "balanced", or "quality".
webpage_policy	Optional structured OpenWebpage policy list with keys: max_open_calls, host_cooldown_seconds, blocked_host_ttl_seconds, open_only_if_score_ge, and parallel_open_limit.
query_templates	Optional named list/dict of backend query template overrides used when orchestration components generate follow-up searches.
verbose	Print progress messages (default: FALSE)
allow_read_webpages	If TRUE, allows the agent to open and read full webpages (HTML/text) via the OpenWebpage tool. Disabled by default.
webpage_relevance_mode	Relevance selection for opened webpages. One of: "auto" (default), "lexical", "embeddings". When "embeddings" or "auto" with an available provider, the tool uses vector similarity to pick the most relevant excerpts; otherwise it falls back to lexical overlap.
webpage_embedding_provider	Embedding provider to use for relevance. One of: "auto" (default), "openai", "sentence_transformers".
webpage_embedding_model	Embedding model identifier. For OpenAI, defaults to "text-embedding-3-small". For sentence-transformers, use a local model name (e.g., "all-MiniLM-L6-v2").
use_plan_mode	Logical flag. When TRUE, the agent creates and follows a structured execution plan, updating step status during the run.
recursion_limit	Optional maximum number of agent steps. Precedence is: per-call value (this argument), then configured default from initialize_agent() / asa_config(), then mode-specific fallback (memory folding: 100; standard agent: 20).

Details

This function provides the primary interface for running research tasks. For simple text responses, use `output_format = "text"`. For structured outputs, use `output_format = "json"` or specify field names to extract. Full trace information is returned via `raw_output` (and `trace_json` when available) for every invocation. For low-level debugging with full underlying response access, use `output_format = "raw"`.

When temporal filtering is specified, the search tool's time filter is temporarily set for this task and restored afterward. Date hints (after/before) are appended to the prompt to guide the agent's search behavior.

Value

An `asa_result` object with:

- `prompt`: The original prompt
- `message`: The agent's response text

- `parsed`: Parsed output (list for JSON/field extraction, NULL for text/raw)
- `raw_output`: Full agent trace (always included)
- `trace_json`: Structured trace JSON (when available)
- `elapsed_time`: Execution time in minutes
- `status`: "success" or "error"
- `search_tier`: Which search tier was used ("primp", "selenium", etc.)
- `parsing_status`: Validation result (if `expected_fields` provided)
- `execution`: Operational metadata (`thread_id`, `stop_reason`, `status_code`, tool budget counters, `fold_count`, `completion_gate`, `verification_status`) plus diagnostics counters from runtime quality guards, candidate and retrieval telemetry, finalization status, artifact status, and orchestration policy metadata.
- `action_ascii`: High-level ASCII action map derived from the trace (also available at `execution$action_ascii`)
- `action_steps`: Parsed high-level action steps (also available at `execution$action_steps`)
- `action_overall`: High-level action summary lines (also available at `execution$action_overall`)
- `langgraph_step_timings`: Per-node LangGraph timings in minutes (also available at `execution$langgraph_step_`)
- `fold_stats`: Memory folding diagnostics list
- `raw_response`: Full underlying Python response object (for `output_format = "raw"`)

See Also

[initialize_agent](#), [run_task_batch](#), [asa_config](#), [temporal_options](#)

Examples

```
## Not run:
# Initialize agent first
agent <- initialize_agent(backend = "openai", model = "gpt-4.1-mini")

# Simple text query
result <- run_task(
  prompt = "What is the capital of France?",
  output_format = "text",
  agent = agent
)
print(result$message)

# JSON structured output
result <- run_task(
  prompt = "Find information about Albert Einstein and return JSON with
           fields: birth_year, death_year, nationality, field_of_study",
  output_format = "json",
  agent = agent
)
print(result$parsed)

# Full traces are always included in asa_result
result <- run_task(
  prompt = "Search for information",
  agent = agent
)
cat(result$raw_output)
```

```

# Use output_format = "raw" only when you also need raw_response
result <- run_task(
  prompt = "Search for information",
  output_format = "raw",
  agent = agent
)
str(result$raw_response) # Inspect full underlying Python response

# With temporal filtering (past year only)
result <- run_task(
  prompt = "Find recent AI research breakthroughs",
  temporal = temporal_options(time_filter = "y"),
  agent = agent
)

# With date range hint
result <- run_task(
  prompt = "Find tech companies founded recently",
  temporal = list(
    time_filter = "y",
    after = "2020-01-01",
    before = "2024-01-01"
  ),
  agent = agent
)

# Using asa_config for unified configuration
config <- asa_config(
  backend = "openai",
  model = "gpt-4.1-mini",
  temporal = temporal_options(time_filter = "y")
)
result <- run_task("Find recent AI research breakthroughs", config = config)

## End(Not run)

```

run_task_batch

Run Multiple Tasks in Batch

Description

Executes multiple research tasks, optionally in parallel. Includes a circuit breaker that monitors error rates and pauses execution if errors spike, preventing cascading failures.

Usage

```

run_task_batch(
  prompts,
  output_format = "text",
  temporal = NULL,
  config = NULL,
  agent = NULL,

```

```

parallel = FALSE,
workers = NULL,
progress = TRUE,
circuit_breaker = TRUE,
abort_on_trip = FALSE,
field_status = NULL,
budget_state = NULL,
search_budget_limit = NULL,
unknown_after_searches = NULL,
finalize_on_all_fields_resolved = NULL,
orchestration_options = NULL,
performance_profile = NULL,
webpage_policy = NULL,
allow_read_webpages = NULL,
webpage_relevance_mode = NULL,
webpage_embedding_provider = NULL,
webpage_embedding_model = NULL,
recursion_limit = NULL
)

```

Arguments

prompts	Character vector of task prompts, or a data frame with a 'prompt' column
output_format	Expected output format (applies to all tasks)
temporal	Named list for temporal filtering (applies to all tasks). See run_task for details.
config	Optional <code>asa_config</code> object to apply across the batch. When provided, <code>config\$temporal</code> is used if <code>temporal</code> is <code>NULL</code> , and <code>config\$workers</code> is used if <code>workers</code> is <code>NULL</code> . In parallel mode, the config is sent to worker processes so each worker can initialize its own agent session.
agent	An <code>asa_agent</code> object
parallel	Use parallel processing
workers	Number of parallel workers
progress	Show progress messages
circuit_breaker	Enable circuit breaker for error rate monitoring. When enabled, tracks recent error rates and pauses if threshold exceeded. Default <code>TRUE</code> .
abort_on_trip	If <code>TRUE</code> , abort the batch when circuit breaker trips. If <code>FALSE</code> (default), wait for cooldown and continue.
field_status	Optional named list tracking per-field resolution status. Passed through to run_task .
budget_state	Optional list tracking search budget consumption across batch items. Passed through to run_task .
search_budget_limit	Optional integer maximum number of searches per task. Passed through to run_task .
unknown_after_searches	Optional integer threshold: if a field remains unknown after this many searches, mark it resolved. Passed through to run_task .
finalize_on_all_fields_resolved	Optional logical; if <code>TRUE</code> , finalize immediately when all schema fields are resolved. Passed through to run_task .

orchestration_options	Optional named list/dict of generic orchestration options passed through to run_task .
performance_profile	Optional orchestration profile passed through to run_task .
webpage_policy	Optional structured OpenWebpage policy passed through to run_task .
allow_read_webpages	If TRUE, allows the agent to open and read full webpages (HTML/text) via the OpenWebpage tool. Disabled by default.
webpage_relevance_mode	Relevance selection for opened webpages. One of: "auto", "lexical", "embeddings". See run_task .
webpage_embedding_provider	Embedding provider for relevance. See run_task .
webpage_embedding_model	Embedding model identifier. See run_task .
recursion_limit	Optional maximum number of agent steps applied to each task. See run_task .

Value

A list of `asa_result` objects, or if prompts was a data frame, the data frame with result columns added (including per-row operational metadata and full per-row `asa_result` objects via list column `asa_result` and attribute `asa_results`). If circuit breaker aborts, includes attribute `"circuit_breaker_aborted" = TRUE`.

See Also

[run_task](#), [configure_temporal](#)

Examples

```
## Not run:
prompts <- c(
  "What is the population of Tokyo?",
  "What is the population of New York?",
  "What is the population of London?"
)
results <- run_task_batch(prompts, agent = agent)

# With temporal filtering for all tasks
results <- run_task_batch(
  prompts,
  temporal = list(time_filter = "y"),
  agent = agent
)

# Disable circuit breaker
results <- run_task_batch(prompts, agent = agent, circuit_breaker = FALSE)

# Abort on circuit breaker trip
results <- run_task_batch(prompts, agent = agent, abort_on_trip = TRUE)

## End(Not run)
```

search_options

*Create Search Options***Description**

Creates search configuration for controlling DuckDuckGo search behavior, including rate limiting, retry policies, and result limits. These options are used by the 4-tier search fallback system.

Usage

```
search_options(
    max_results = NULL,
    timeout = NULL,
    max_retries = NULL,
    retry_delay = NULL,
    backoff_multiplier = NULL,
    inter_search_delay = NULL,
    humanize_timing = NULL,
    jitter_factor = NULL,
    allow_direct_fallback = NULL,
    selenium_browser_preference = NULL,
    stability_profile = NULL,
    performance_profile = NULL,
    webpage_policy = NULL,
    auto_openwebpage_policy = NULL,
    langgraph_node_retries = NULL,
    langgraph_cache_enabled = NULL,
    finalize_when_all_unresolved_exhausted = NULL,
    field_attempt_budget_mode = NULL,
    allow_read_webpages = NULL,
    webpage_relevance_mode = NULL,
    webpage_heuristic_profile = NULL,
    webpage_embedding_provider = NULL,
    webpage_embedding_model = NULL,
    webpage_timeout = NULL,
    webpage_max_bytes = NULL,
    webpage_max_chars = NULL,
    webpage_max_chunks = NULL,
    webpage_chunk_chars = NULL,
    webpage_embedding_api_base = NULL,
    webpage_prefilter_k = NULL,
    webpage_use_mmr = NULL,
    webpage_mmr_lambda = NULL,
    webpage_cache_enabled = NULL,
    webpage_cache_max_entries = NULL,
    webpage_cache_max_text_chars = NULL,
    webpage_blocked_cache_ttl_sec = NULL,
    webpage_blocked_cache_max_entries = NULL,
    webpage_blocked_probe_bytes = NULL,
    webpage_blocked_detect_on_200 = NULL,
    webpage_blocked_body_scan_bytes = NULL,
```

```

webpage_pdf_enabled = NULL,
webpage_pdf_timeout = NULL,
webpage_pdf_max_bytes = NULL,
webpage_pdf_max_pages = NULL,
webpage_pdf_max_text_chars = NULL,
webpage_user_agent = NULL,
wikidata_template_path = NULL,
wiki_top_k_results = NULL,
wiki_doc_content_chars_max = NULL,
search_doc_content_chars_max = NULL
)

```

Arguments

max_results	Maximum number of search results to return per query. Higher values provide more context but increase latency. Default: 10.
timeout	Timeout in seconds for individual search requests. Applies to each tier attempt separately. Default: 30.
max_retries	Maximum number of retry attempts when a search tier fails. After exhausting retries, the system falls back to the next tier. Default: 3.
retry_delay	Initial delay in seconds before the first retry. Subsequent retries use exponential backoff. Default: 2.
backoff_multiplier	Multiplier for exponential backoff between retries. E.g., with retry_delay=2 and multiplier=1.5, delays are 2s, 3s, 4.5s. Default: 1.5.
inter_search_delay	Minimum delay in seconds between consecutive searches. Helps avoid rate limiting from search providers. Default: 1.5.
humanize_timing	Whether to humanize search timing with non-uniform delays. Default uses ASA_HUMANIZE_TIMING.
jitter_factor	Random jitter factor applied to humanized timing and retry spacing. Default uses ASA_JITTER_FACTOR.
allow_direct_fallback	Whether to allow direct fallback requests in the search transport. Default: FALSE.
selenium_browser_preference	Selenium browser engine preference order. One of "firefox_first" (default) or "chrome_first".
stability_profile	High-level search stability profile. One of "deterministic", "balanced", or "stealth_first" (default). This controls default anti-detection and orchestration behavior.
performance_profile	Orchestration profile optimized for one of: "latency", "balanced", or "quality".
webpage_policy	Structured OpenWebpage policy list with keys: max_open_calls, host_cooldown_seconds, blocked_host_ttl_seconds, open_only_if_score_ge, and parallel_open_limit.
auto_openwebpage_policy	Automatic OpenWebpage follow-up policy. Use "auto" for profile-driven automatic opening or "off" to disable it. Defaults to "off" for the deterministic profile and "auto" otherwise.

langgraph_node_retries	Whether to enable LangGraph node-level retries in agent graph construction. Default: TRUE.
langgraph_cache_enabled	Whether to enable LangGraph node caching. Default: FALSE.
finalize_when_all_unresolved_exhausted	Optional override for orchestration finalizer behavior. When TRUE, finalize once all unresolved fields have exhausted their search-attempt budget.
field_attempt_budget_mode	Optional field-attempt budget mode passed to orchestration field_resolver. One of "strict_cap" or "soft_cap".
allow_read_webpages	If TRUE, allows the agent to open and read full webpages (HTML/text) via the OpenWebpage tool. Disabled by default.
webpage_relevance_mode	Relevance selection for opened webpages. One of: "auto", "lexical", "embeddings".
webpage_heuristic_profile	Heuristic profile used when annotating links from opened webpages. Must be "generic" (task-agnostic default).
webpage_embedding_provider	Embedding provider for relevance. One of: "auto", "openai", "sentence_transformers".
webpage_embedding_model	Embedding model identifier for relevance.
webpage_timeout	Timeout in seconds for OpenWebpage fetches and (when used) relevance embeddings.
webpage_max_bytes	Maximum bytes to download per page (hard cap).
webpage_max_chars	Maximum characters returned from an opened page (hard cap on tool output).
webpage_max_chunks	Maximum number of relevant excerpts to include from a page.
webpage_chunk_chars	Approximate size (in characters) of each excerpt.
webpage_embedding_api_base	Optional OpenAI-compatible base URL for embeddings. If NULL, uses OPENAI_API_BASE or https://api.openai.com/v1.
webpage_prefilter_k	Optional lexical prefilter size before embedding relevance selection.
webpage_use_mmr	Whether to apply maximal marginal relevance (MMR) for more diverse excerpts.
webpage_mmr_lambda	MMR tradeoff between relevance (1.0) and diversity (0.0).
webpage_cache_enabled	Enable per-run caching of opened pages.
webpage_cache_max_entries	Max cached page entries per run.

webpage_cache_max_text_chars	Max characters of extracted page text to store per cached page (separate from webpage_max_chars, which caps tool output).
webpage_blocked_cache_ttl_sec	TTL in seconds for cached blocked URL responses (HTTP 403/429 and anti-bot pages).
webpage_blocked_cache_max_entries	Max blocked-URL cache entries per run.
webpage_blocked_probe_bytes	Max bytes to inspect when classifying a response as blocked.
webpage_blocked_detect_on_200	If TRUE, classify anti-bot interstitials served with HTTP 200 as blocked responses.
webpage_blocked_body_scan_bytes	Max response bytes scanned on HTTP 200 pages for anti-bot marker detection.
webpage_pdf_enabled	If TRUE, OpenWebpage attempts PDF extraction via pdftotext.
webpage_pdf_timeout	Timeout in seconds for pdftotext.
webpage_pdf_max_bytes	Max PDF bytes to download before extraction.
webpage_pdf_max_pages	Max pages passed to pdftotext.
webpage_pdf_max_text_chars	Max extracted PDF characters kept before relevance selection/output truncation.
webpage_user_agent	User-Agent string used for webpage fetches.
wikidata_template_path	Optional path to a Wikidata template JSON file. When provided, Wikidata known-entity catalogs are loaded from this file. If NULL (default), no task-specific template catalog is auto-loaded.
wiki_top_k_results	Number of Wikipedia search results to fetch per query (default: 5). Higher values may provide more coverage at the cost of latency/noise.
wiki_doc_content_chars_max	Maximum number of characters to include from each Wikipedia page/summary in the Wikipedia tool output (default: 1000).
search_doc_content_chars_max	Maximum number of characters to include from DuckDuckGo Search tool outputs (default: 500). This caps the text returned to the LLM per Search tool invocation.

Details

The search system uses a 4-tier fallback architecture:

1. **PRIMP**: HTTP/2 with browser TLS fingerprint
2. **Selenium**: Headless browser for JS-rendered content
3. **DDGS**: Standard ddgs Python library

4. **Requests:** Raw POST to DuckDuckGo HTML endpoint

The retry/backoff settings apply within each tier. If all retries are exhausted, the system automatically falls back to the next tier. Selenium browser initialization defaults to "firefox_first" and falls back to Chrome engines when Firefox is unavailable.

Value

An object of class `asa_search`

See Also

[asa_config](#), [configure_search](#)

Examples

```
## Not run:
# Default settings
search <- search_options()

# More aggressive settings for faster searches
search <- search_options(
  max_results = 5,
  timeout = 10,
  max_retries = 2
)

# Conservative settings for rate-limited environments
search <- search_options(
  inter_search_delay = 2.0,
  max_retries = 5,
  backoff_multiplier = 2.0
)

# Use with asa_config
config <- asa_config(
  backend = "openai",
  search = search_options(max_results = 15)
)

## End(Not run)
```

summary.asa_agent

Summary Method for asa_agent Objects

Description

Summary Method for `asa_agent` Objects

Usage

```
## S3 method for class 'asa_agent'
summary(object, ...)
```

Arguments

object	An asa_agent object
...	Additional arguments (ignored)

Value

Invisibly returns a summary list

`summary.asa_audit_result`

Summary Method for asa_audit_result Objects

Description

Summary Method for asa_audit_result Objects

Usage

```
## S3 method for class 'asa_audit_result'  
summary(object, ...)
```

Arguments

object	An asa_audit_result object
...	Additional arguments (ignored)

Value

Invisibly returns a summary list

`summary.asa_benchmark` *Summary Method for asa_benchmark Objects*

Description

Summary Method for asa_benchmark Objects

Usage

```
## S3 method for class 'asa_benchmark'  
summary(object, ...)
```

Arguments

object	An asa_benchmark object
...	Additional arguments (ignored)

Value

Invisibly returns a benchmark summary list

`summary.asa_enumerate_result`*Summary Method for asa_enumerate_result Objects*

Description

Summary Method for asa_enumerate_result Objects

Usage

```
## S3 method for class 'asa_enumerate_result'  
summary(object, ...)
```

Arguments

<code>object</code>	An asa_enumerate_result object
<code>...</code>	Additional arguments (ignored)

Value

Invisibly returns a summary list

`summary.asa_evaluation_result`*Summary Method for asa_evaluation_result Objects*

Description

Summary Method for asa_evaluation_result Objects

Usage

```
## S3 method for class 'asa_evaluation_result'  
summary(object, ...)
```

Arguments

<code>object</code>	An asa_evaluation_result object
<code>...</code>	Additional arguments (ignored)

Value

Invisibly returns an evaluation summary list

summary.asa_response *Summary Method for asa_response Objects*

Description

Summary Method for asa_response Objects

Usage

```
## S3 method for class 'asa_response'  
summary(object, show_trace = FALSE, ...)
```

Arguments

object	An asa_response object
show_trace	Include full trace in output
...	Additional arguments (ignored)

Value

Invisibly returns a summary list

summary.asa_result *Summary Method for asa_result Objects*

Description

Summary Method for asa_result Objects

Usage

```
## S3 method for class 'asa_result'  
summary(object, ...)
```

Arguments

object	An asa_result object
...	Additional arguments (ignored)

Value

Invisibly returns a summary list

temporal_options	Create Temporal Filtering Options
------------------	-----------------------------------

Description

Creates a temporal filtering configuration for constraining search results by date. Supports DuckDuckGo time filters, date ranges, and strict verification modes.

Usage

```
temporal_options(  
  time_filter = NULL,  
  after = NULL,  
  before = NULL,  
  strictness = "best_effort",  
  use_wayback = FALSE  
)
```

Arguments

time_filter	DuckDuckGo time filter: "d" (day), "w" (week), "m" (month), "y" (year), or NULL for no filter
after	ISO 8601 date string (e.g., "2020-01-01") - results after this date
before	ISO 8601 date string (e.g., "2024-01-01") - results before this date
strictness	Verification level: "best_effort" (default) or "strict"
use_wayback	Use Wayback Machine for strict pre-date guarantees

Details

Temporal filtering can operate at different levels:

- **time_filter**: DuckDuckGo native filter (fast, approximate)
- **after/before**: Date hints appended to prompts
- **strict**: Post-hoc verification of result dates
- **use_wayback**: Uses Internet Archive for guaranteed historical data

Value

An object of class `asa_temporal`

See Also

[asa_config](#), [run_task](#)

Examples

```
## Not run:
# Past year only
temporal <- temporal_options(time_filter = "y")

# Specific date range
temporal <- temporal_options(
  after = "2020-01-01",
  before = "2024-01-01"
)

# Strict historical verification
temporal <- temporal_options(
  before = "2015-01-01",
  strictness = "strict",
  use_wayback = TRUE
)

## End(Not run)
```

tor_options

*Tor Options***Description**

Configure shared Tor exit tracking for healthier circuit rotation.

Usage

```
tor_options(
  registry_path = NULL,
  dirty_tor_exists = ASA_TOR_REGISTRY_ENABLED,
  bad_ttl = ASA_TOR_BAD_TTL,
  good_ttl = ASA_TOR_GOOD_TTL,
  overuse_threshold = ASA_TOR_OVERUSE_THRESHOLD,
  overuse_decay = ASA_TOR_OVERUSE_DECAY,
  max_rotation_attempts = ASA_TOR_MAX_ROTATION_ATTEMPTS,
  ip_cache_ttl = ASA_TOR_IP_CACHE_TTL
)
```

Arguments

registry_path	Path to the shared SQLite registry file (default: user cache).
dirty_tor_exists	Enable the registry (tracks good/bad/overused exits).
bad_ttl	Seconds to keep a bad/tainted exit before reuse (default: 3600).
good_ttl	Seconds to treat an exit as good before refreshing (default: 1800).
overuse_threshold	Max recent uses before a good exit is considered overloaded.
overuse_decay	Window (seconds) for overuse counting before decaying.

`max_rotation_attempts` Max attempts to find a clean exit before giving up.
`ip_cache_ttl` Seconds to cache exit IP lookups.

Value

An object of class `asa_tor`

`write_csv.asa_enumerate_result`
Write `asa_enumerate_result` to CSV

Description

Write `asa_enumerate_result` to CSV

Usage

```
write_csv.asa_enumerate_result(x, file, include_provenance = FALSE, ...)
```

Arguments

`x` An `asa_enumerate_result` object
`file` Path to output CSV file
`include_provenance` Include provenance as additional columns
`...` Additional arguments passed to `write.csv`

Value

Invisibly returns the file path

Index

as.data.frame.asa_audit_result, 3
as.data.frame.asa_enumerate_result, 3
as.data.frame.asa_evaluation_result, 4
as.data.frame.asa_result, 4
asa_agent, 5
asa_audit, 5
asa_audit_result, 7
asa_config, 8, 50, 58, 62
asa_enumerate, 10, 26, 34
asa_enumerate_result, 14
asa_evaluation_result, 15
asa_response, 16
asa_result, 18

benchmark_from_df, 19
benchmark_from_yaml, 20
build_backend, 20
build_prompt, 22

check_backend, 22
configure_search, 23, 58
configure_search_logging, 24
configure_temporal, 25, 53
configure_tor_registry, 26
create_benchmark, 27

evaluate_agent, 28
extract_agent_results, 29
extract_search_snippets, 30
extract_search_tiers, 30
extract_urls, 31
extract_wikipedia_content, 32

get_agent, 32
get_tor_ip, 33

initialize_agent, 13, 33, 46, 48, 50
is_tor_running, 37

list_benchmarks, 37
load_benchmark, 38

print.asa_agent, 38
print.asa_audit_result, 39
print.asa_benchmark, 39
print.asa_config, 40
print.asa_enumerate_result, 40
print.asa_evaluation_result, 41
print.asa_response, 41
print.asa_result, 42
print.asa_search, 42
print.asa_temporal, 43
print.asa_tor, 43
process_outputs, 44

reset_agent, 44
rotate_tor_circuit, 45
run_direct_task, 46
run_task, 9, 13, 18, 26, 34, 36, 47, 52, 53, 62
run_task_batch, 9, 36, 50, 51

search_options, 10, 34, 54
summary.asa_agent, 58
summary.asa_audit_result, 59
summary.asa_benchmark, 59
summary.asa_enumerate_result, 60
summary.asa_evaluation_result, 60
summary.asa_response, 61
summary.asa_result, 61

temporal_options, 10, 50, 62
tor_options, 35, 63

write_csv.asa_enumerate_result, 64