

# Package ‘asa’

December 26, 2025

**Title** AI Search Agent for Large-Scale Research Automation

**Version** 0.1.0

**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', and 'xAI'.

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

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

**Depends** R (>= 4.0.0)

**License** GPL-3

**Encoding** UTF-8

**Imports** reticulate (>= 1.28), jsonlite, rlang, digest

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

**VignetteBuilder** knitr

**RoxygenNote** 7.3.3

**Config/testthat/edition** 3

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

**NeedsCompilation** no

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

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

## Contents

|                                   |   |
|-----------------------------------|---|
| asa-package . . . . .             | 3 |
| .build_trace . . . . .            | 4 |
| .close_http_clients . . . . .     | 5 |
| .create_agent . . . . .           | 5 |
| .create_http_clients . . . . .    | 6 |
| .create_llm . . . . .             | 6 |
| .create_research_config . . . . . | 7 |
| .create_research_graph . . . . .  | 7 |

|  |    |
|--|----|
| .create_tools . . . . .                      | 7  |
| .extract_fields . . . . .                    | 8  |
| .extract_json_from_trace . . . . .           | 8  |
| .extract_json_object . . . . .               | 8  |
| .extract_response_text . . . . .             | 9  |
| .get_extdata_path . . . . .                  | 9  |
| .get_local_ip . . . . .                      | 9  |
| .get_python_path . . . . .                   | 10 |
| .handle_response_issues . . . . .            | 10 |
| .import_python_packages . . . . .            | 10 |
| .import_research_modules . . . . .           | 10 |
| .invoke_memory_folding_agent . . . . .       | 11 |
| .invoke_standard_agent . . . . .             | 11 |
| .is_initialized . . . . .                    | 11 |
| .normalize_schema . . . . .                  | 11 |
| .parse_json_response . . . . .               | 12 |
| .process_research_results . . . . .          | 12 |
| .resume_research . . . . .                   | 12 |
| .run_research . . . . .                      | 12 |
| .run_research_with_progress . . . . .        | 13 |
| .save_checkpoint . . . . .                   | 13 |
| .stop_validation . . . . .                   | 13 |
| .validate_asa_agent . . . . .                | 14 |
| .validate_asa_response . . . . .             | 14 |
| .validate_asa_result . . . . .               | 14 |
| .validate_build_backend . . . . .            | 15 |
| .validate_build_prompt . . . . .             | 15 |
| .validate_choice . . . . .                   | 15 |
| .validate_conda_env . . . . .                | 16 |
| .validate_configure_search . . . . .         | 16 |
| .validate_consistency . . . . .              | 17 |
| .validate_dataframe . . . . .                | 17 |
| .validate_initialize_agent . . . . .         | 18 |
| .validate_logical . . . . .                  | 18 |
| .validate_positive . . . . .                 | 19 |
| .validate_process_outputs . . . . .          | 19 |
| .validate_proxy_url . . . . .                | 19 |
| .validate_range . . . . .                    | 20 |
| .validate_required . . . . .                 | 20 |
| .validate_research_inputs . . . . .          | 21 |
| .validate_run_agent . . . . .                | 21 |
| .validate_run_task . . . . .                 | 21 |
| .validate_run_task_batch . . . . .           | 22 |
| .validate_s3_class . . . . .                 | 22 |
| .validate_string . . . . .                   | 23 |
| .validate_string_vector . . . . .            | 23 |
| as.data.frame.asa_enumerate_result . . . . . | 24 |
| as.data.frame.asa_result . . . . .           | 24 |
| asa_agent . . . . .                          | 25 |
| asa_enumerate . . . . .                      | 25 |
| asa_enumerate_result . . . . .               | 28 |
| asa_response . . . . .                       | 29 |

|  |    |
|--|----|
| asa_result . . . . .                     | 30 |
| build_backend . . . . .                  | 30 |
| build_prompt . . . . .                   | 31 |
| check_backend . . . . .                  | 32 |
| clean_whitespace . . . . .               | 33 |
| configure_search . . . . .               | 33 |
| configure_search_logging . . . . .       | 34 |
| decode_html . . . . .                    | 35 |
| extract_agent_results . . . . .          | 36 |
| extract_search_snippets . . . . .        | 36 |
| extract_search_tiers . . . . .           | 37 |
| extract_urls . . . . .                   | 38 |
| extract_wikipedia_content . . . . .      | 38 |
| format_duration . . . . .                | 39 |
| get_agent . . . . .                      | 39 |
| get_tor_ip . . . . .                     | 40 |
| initialize_agent . . . . .               | 40 |
| is_tor_running . . . . .                 | 42 |
| json_escape . . . . .                    | 43 |
| print.asa_agent . . . . .                | 43 |
| print.asa_enumerate_result . . . . .     | 44 |
| print.asa_response . . . . .             | 44 |
| print.asa_result . . . . .               | 45 |
| print2 . . . . .                         | 45 |
| process_outputs . . . . .                | 46 |
| reset_agent . . . . .                    | 46 |
| rotate_tor_circuit . . . . .             | 47 |
| run_agent . . . . .                      | 47 |
| run_agent_batch . . . . .                | 48 |
| run_task . . . . .                       | 49 |
| run_task_batch . . . . .                 | 50 |
| safe_json_parse . . . . .                | 51 |
| summary.asa_agent . . . . .              | 52 |
| summary.asa_enumerate_result . . . . .   | 52 |
| summary.asa_response . . . . .           | 53 |
| summary.asa_result . . . . .             | 53 |
| truncate_string . . . . .                | 54 |
| write_csv.asa_enumerate_result . . . . . | 54 |

**Index****55**

asa-package

*asa: AI Search Agent for Large-Scale Research Automation***Description**

The asa package provides an LLM-powered research agent for performing AI search tasks at large scales using web search capabilities.

The agent uses a ReAct (Reasoning + Acting) pattern implemented via LangGraph, with tools for searching DuckDuckGo and Wikipedia. It supports multiple LLM backends (OpenAI, Groq, xAI) and implements DeepAgent-style memory folding for managing long conversations.

## Main Functions

- [build\\_backend](#): Set up the Python conda environment
- [initialize\\_agent](#): Initialize the search agent
- [run\\_agent](#): Run the agent with a custom prompt
- [run\\_task](#): Run a structured task with the agent
- [run\\_task\\_batch](#): Run multiple tasks in batch

## Configuration

The package requires a Python environment with LangChain and related packages. Use [build\\_backend](#) to create this environment automatically.

For anonymous searching, the package can use Tor as a SOCKS5 proxy. Install Tor via `brew install tor` (macOS) and start it with `brew services start tor`.

## Author(s)

**Maintainer:** Connor Jerzak <connor.jerzak@gmail.com> ([ORCID](#))

## See Also

Useful links:

- <https://github.com/cjerzak/asa-software>
- Report bugs at <https://github.com/cjerzak/asa-software/issues>

---

*.build\_trace**Build Trace from Raw Response*

---

## Description

Build Trace from Raw Response

## Usage

```
.build_trace(raw_response)
```

---

|                                  |                           |
|----------------------------------|---------------------------|
| <code>.close_http_clients</code> | <i>Close HTTP Clients</i> |
|----------------------------------|---------------------------|

---

**Description**

Safely closes the synchronous httpx client to prevent resource leaks. This is called automatically by `reset_agent()` and when reinitializing.

**Usage**

```
.close_http_clients()
```

**Details**

Note: We no longer create or manage async clients from R (R-CRIT-001 fix). LangChain manages its own async client lifecycle internally.

**Value**

Invisibly returns NULL

---

|                            |                                   |
|----------------------------|-----------------------------------|
| <code>.create_agent</code> | <i>Create the LangGraph Agent</i> |
|----------------------------|-----------------------------------|

---

**Description**

Create the LangGraph Agent

**Usage**

```
.create_agent(  
  llm,  
  tools,  
  use_memory_folding,  
  memory_threshold,  
  memory_keep_recent  
)
```

**Arguments**

|                                 |                               |
|---------------------------------|-------------------------------|
| <code>llm</code>                | LLM instance                  |
| <code>tools</code>              | List of tools                 |
| <code>use_memory_folding</code> | Whether to use memory folding |
| <code>memory_threshold</code>   | Messages before folding       |
| <code>memory_keep_recent</code> | Messages to keep              |

---

`.create_http_clients`    *Create HTTP Client for API Calls*

---

### Description

Creates a synchronous httpx client for LLM API calls. Note: We intentionally do NOT create an async client. LangChain/OpenAI SDK creates its own async client internally when needed (for async operations). This avoids R-CRIT-001 where async client cleanup was unreliable from R since `aclose()` requires an async context.

### Usage

```
.create_http_clients(proxy, timeout)
```

### Arguments

|                      |                    |
|----------------------|--------------------|
| <code>proxy</code>   | Proxy URL or NULL  |
| <code>timeout</code> | Timeout in seconds |

### Value

A list with 'sync' client (async is NULL, letting LangChain manage it)

---

`.create_llm`    *Create LLM Instance*

---

### Description

Create LLM Instance

### Usage

```
.create_llm(backend, model, clients, rate_limit)
```

### Arguments

|                         |                           |
|-------------------------|---------------------------|
| <code>backend</code>    | Backend name              |
| <code>model</code>      | Model identifier          |
| <code>clients</code>    | HTTP clients (for OpenAI) |
| <code>rate_limit</code> | Requests per second       |

---

.create\_research\_config

*Create Research Configuration*

---

### Description

Create Research Configuration

### Usage

.create\_research\_config(max\_workers, max\_rounds, budget, stop\_policy, sources)

---

.create\_research\_graph

*Create Research Graph*

---

### Description

Create Research Graph

### Usage

.create\_research\_graph(agent, config\_dict)

---

.create\_tools

*Create Search Tools*

---

### Description

Create Search Tools

### Usage

.create\_tools(proxy)

### Arguments

proxy                      Proxy URL or NULL

---

|                 |  |
|-----------------|--|
| .extract_fields | <i>Extract Specific Fields from Response</i> |
|-----------------|--|

---

**Description**

Extract Specific Fields from Response

**Usage**

.extract\_fields(text, fields)

**Arguments**

|        |  |
|--------|--|
| text   | Response text                              |
| fields | Character vector of field names to extract |

---

|                          |                                       |
|--------------------------|---------------------------------------|
| .extract_json_from_trace | <i>Extract JSON from Agent Traces</i> |
|--------------------------|---------------------------------------|

---

**Description**

Internal function to extract JSON data from raw agent traces.

**Usage**

.extract\_json\_from\_trace(text)

**Arguments**

|      |                |
|------|----------------|
| text | Raw trace text |
|------|----------------|

**Value**

Parsed JSON data as a list, or NULL if no JSON found

---

|                      |                                      |
|----------------------|--------------------------------------|
| .extract_json_object | <i>Extract JSON Object from Text</i> |
|----------------------|--------------------------------------|

---

**Description**

Extract JSON Object from Text

**Usage**

.extract\_json\_object(text)

**Arguments**

|      |               |
|------|---------------|
| text | Response text |
|------|---------------|

---

.extract\_response\_text

*Extract Response Text from Raw Response*

---

### Description

Extract Response Text from Raw Response

### Usage

.extract\_response\_text(raw\_response, backend)

---

.get\_extdata\_path

*Get External Data Path*

---

### Description

Returns the path to the package's external data directory.

### Usage

.get\_extdata\_path(filename = NULL)

### Arguments

filename            Optional filename within extdata directory

### Value

Character string with the path

---

.get\_local\_ip

*Get Local IP Address (Cross-Platform)*

---

### Description

Returns the local IP address for use with Exo backend. Works on Windows, macOS, and Linux.

### Usage

.get\_local\_ip()

### Value

Character string with the local IP address, or "127.0.0.1" on failure.

---

|                               |                                       |
|-------------------------------|---------------------------------------|
| <code>.get_python_path</code> | <i>Get Package Python Module Path</i> |
|-------------------------------|---------------------------------------|

---

**Description**

Returns the path to the Python modules shipped with the package.

**Usage**

```
.get_python_path()
```

**Value**

Character string with the path to inst/python

---

|                                      |   |
|--------------------------------------|---|
| <code>.handle_response_issues</code> | <i>Handle Response Issues (Rate Limiting, Timeouts)</i> |
|--------------------------------------|---|

---

**Description**

Handle Response Issues (Rate Limiting, Timeouts)

**Usage**

```
.handle_response_issues(trace, verbose)
```

---

|                                      |  |
|--------------------------------------|--|
| <code>.import_python_packages</code> | <i>Import Required Python Packages</i> |
|--------------------------------------|--|

---

**Description**

Import Required Python Packages

**Usage**

```
.import_python_packages()
```

---

|                                       |                                       |
|---------------------------------------|---------------------------------------|
| <code>.import_research_modules</code> | <i>Import Research Python Modules</i> |
|---------------------------------------|---------------------------------------|

---

**Description**

Import Research Python Modules

**Usage**

```
.import_research_modules()
```

---

|   |                                    |
|---|------------------------------------|
| <code>.invoke_memory_folding_agent</code> | <i>Invoke Memory Folding Agent</i> |
|---|------------------------------------|

---

**Description**

Invoke Memory Folding Agent

**Usage**

```
.invoke_memory_folding_agent(python_agent, prompt, recursion_limit)
```

---

|                                     |                              |
|-------------------------------------|------------------------------|
| <code>.invoke_standard_agent</code> | <i>Invoke Standard Agent</i> |
|-------------------------------------|------------------------------|

---

**Description**

Invoke Standard Agent

**Usage**

```
.invoke_standard_agent(python_agent, prompt, recursion_limit)
```

---

|                              |  |
|------------------------------|--|
| <code>.is_initialized</code> | <i>Check if ASA Agent is Initialized</i> |
|------------------------------|--|

---

**Description**

Check if ASA Agent is Initialized

**Usage**

```
.is_initialized()
```

**Value**

Logical indicating if the agent has been initialized

---

|                                |                               |
|--------------------------------|-------------------------------|
| <code>.normalize_schema</code> | <i>Normalize Schema Input</i> |
|--------------------------------|-------------------------------|

---

**Description**

Normalize Schema Input

**Usage**

```
.normalize_schema(schema, query, verbose)
```

---

|                                   |                            |
|-----------------------------------|----------------------------|
| <code>.parse_json_response</code> | <i>Parse JSON Response</i> |
|-----------------------------------|----------------------------|

---

**Description**

Parse JSON Response

**Usage**

`.parse_json_response(response_text)`

**Arguments**

`response_text`    Response text from agent

---

|  |                                 |
|--|---------------------------------|
| <code>.process_research_results</code> | <i>Process Research Results</i> |
|--|---------------------------------|

---

**Description**

Process Research Results

**Usage**

`.process_research_results(result, schema_dict, include_provenance)`

---

|                               |  |
|-------------------------------|--|
| <code>.resume_research</code> | <i>Resume Research from Checkpoint</i> |
|-------------------------------|--|

---

**Description**

Resume Research from Checkpoint

**Usage**

`.resume_research(checkpoint_file, verbose)`

---

|                            |                                     |
|----------------------------|-------------------------------------|
| <code>.run_research</code> | <i>Run Research (Non-Streaming)</i> |
|----------------------------|-------------------------------------|

---

**Description**

Run Research (Non-Streaming)

**Usage**

`.run_research(graph, query, schema_dict, config_dict)`

---

|  |   |
|--|---|
| <code>.run_research_with_progress</code> | <i>Run Research with Progress Updates</i> |
|--|---|

---

**Description**

Run Research with Progress Updates

**Usage**

```
.run_research_with_progress(  
    graph,  
    query,  
    schema_dict,  
    config_dict,  
    checkpoint_file,  
    verbose  
)
```

---

|                               |                        |
|-------------------------------|------------------------|
| <code>.save_checkpoint</code> | <i>Save Checkpoint</i> |
|-------------------------------|------------------------|

---

**Description**

Save Checkpoint

**Usage**

```
.save_checkpoint(result, query, schema_dict, config_dict, checkpoint_file)
```

---

|                               |   |
|-------------------------------|---|
| <code>.stop_validation</code> | <i>Stop with Formatted Validation Error</i> |
|-------------------------------|---|

---

**Description**

Creates a standardized error message with Got/Fix sections.

**Usage**

```
.stop_validation(param_name, requirement, actual = NULL, fix = NULL)
```

**Arguments**

|                          |   |
|--------------------------|---|
| <code>param_name</code>  | Name of the parameter that failed validation          |
| <code>requirement</code> | What the parameter should be                          |
| <code>actual</code>      | What was actually received (optional, auto-formatted) |
| <code>fix</code>         | Actionable fix suggestion                             |

---

`.validate_asa_agent`     *Validate S3 Constructor: asa\_agent*

---

### **Description**

Validate S3 Constructor: asa\_agent

### **Usage**

```
.validate_asa_agent(python_agent, backend, model, config)
```

---

`.validate_asa_response`  
                          *Validate S3 Constructor: asa\_response*

---

### **Description**

Validate S3 Constructor: asa\_response

### **Usage**

```
.validate_asa_response(  
    message,  
    status_code,  
    raw_response,  
    trace,  
    elapsed_time,  
    fold_count,  
    prompt  
)
```

---

`.validate_asa_result`     *Validate S3 Constructor: asa\_result*

---

### **Description**

Validate S3 Constructor: asa\_result

### **Usage**

```
.validate_asa_result(prompt, message, parsed, raw_output, elapsed_time, status)
```

---

.validate\_build\_backend

*Validate build\_backend() Parameters*

---

### Description

Validate build\_backend() Parameters

### Usage

.validate\_build\_backend(conda\_env, conda, python\_version)

---

.validate\_build\_prompt

*Validate build\_prompt() Parameters*

---

### Description

Validate build\_prompt() Parameters

### Usage

.validate\_build\_prompt(template)

---

.validate\_choice

*Validate Choice from Set*

---

### Description

Validate Choice from Set

### Usage

.validate\_choice(x, param\_name, choices)

### Arguments

|            |                        |
|------------|------------------------|
| x          | Value to check         |
| param_name | Name for error message |
| choices    | Valid choices          |

---

|                                  |  |
|----------------------------------|--|
| <code>.validate_conda_env</code> | <i>Validate Conda Environment Name</i> |
|----------------------------------|--|

---

**Description**

Validate Conda Environment Name

**Usage**

```
.validate_conda_env(x, param_name)
```

**Arguments**

|                         |                        |
|-------------------------|------------------------|
| <code>x</code>          | Value to check         |
| <code>param_name</code> | Name for error message |

---

|   |   |
|---|---|
| <code>.validate_configure_search</code> | <i>Validate configure_search() Parameters</i> |
|---|---|

---

**Description**

Validate configure\_search() Parameters

**Usage**

```
.validate_configure_search(  
    max_results,  
    timeout,  
    max_retries,  
    retry_delay,  
    backoff_multiplier,  
    captcha_backoff_base,  
    page_load_wait,  
    inter_search_delay,  
    conda_env  
)
```

---

.validate\_consistency    *Validate Logical Consistency Between Parameters*

---

### Description

Validate Logical Consistency Between Parameters

### Usage

```
.validate_consistency(condition, message, fix)
```

### Arguments

|           |                                     |
|-----------|-------------------------------------|
| condition | Condition that must be TRUE         |
| message   | Error message if condition is FALSE |
| fix       | How to fix the issue                |

---

.validate\_dataframe    *Validate Data Frame with Required Columns*

---

### Description

Validate Data Frame with Required Columns

### Usage

```
.validate_dataframe(x, param_name, required_cols = NULL)
```

### Arguments

|               |                                  |
|---------------|----------------------------------|
| x             | Value to check                   |
| param_name    | Name for error message           |
| required_cols | Required column names (optional) |

---

`.validate_initialize_agent`

*Validate initialize\_agent() Parameters*

---

**Description**

Validate initialize\_agent() Parameters

**Usage**

```
.validate_initialize_agent(  
    backend,  
    model,  
    conda_env,  
    proxy,  
    use_memory_folding,  
    memory_threshold,  
    memory_keep_recent,  
    rate_limit,  
    timeout,  
    verbose  
)
```

---

`.validate_logical`

*Validate Boolean*

---

**Description**

Validate Boolean

**Usage**

```
.validate_logical(x, param_name)
```

**Arguments**

- |            |                        |
|------------|------------------------|
| x          | Value to check         |
| param_name | Name for error message |

---

|                    |                                 |
|--------------------|---------------------------------|
| .validate_positive | <i>Validate Positive Number</i> |
|--------------------|---------------------------------|

---

### Description

Validate Positive Number

### Usage

```
.validate_positive(x, param_name, allow_zero = FALSE, integer_only = FALSE)
```

### Arguments

|              |   |
|--------------|---|
| x            | Value to check                          |
| param_name   | Name for error message                  |
| allow_zero   | Allow zero values (default: FALSE)      |
| integer_only | Require integer values (default: FALSE) |

---

|                           |  |
|---------------------------|--|
| .validate_process_outputs | <i>Validate process_outputs() Parameters</i> |
|---------------------------|--|

---

### Description

Validate process\_outputs() Parameters

### Usage

```
.validate_process_outputs(df, parallel, workers)
```

---

|                     |   |
|---------------------|---|
| .validate_proxy_url | <i>Validate URL Format (SOCKS5 Proxy)</i> |
|---------------------|---|

---

### Description

Validate URL Format (SOCKS5 Proxy)

### Usage

```
.validate_proxy_url(x, param_name)
```

### Arguments

|            |   |
|------------|---|
| x          | Value to check (NULL is valid = no proxy) |
| param_name | Name for error message                    |

---

|                              |                       |
|------------------------------|-----------------------|
| <code>.validate_range</code> | <i>Validate Range</i> |
|------------------------------|-----------------------|

---

**Description**

Validate Range

**Usage**

```
.validate_range(x, param_name, min = NULL, max = NULL)
```

**Arguments**

|                         |   |
|-------------------------|---|
| <code>x</code>          | Value to check (must already be validated as numeric) |
| <code>param_name</code> | Name for error message                                |
| <code>min</code>        | Minimum allowed value (optional)                      |
| <code>max</code>        | Maximum allowed value (optional)                      |

---

|                                 |  |
|---------------------------------|--|
| <code>.validate_required</code> | <i>Validate Required Argument Presence</i> |
|---------------------------------|--|

---

**Description**

Validate Required Argument Presence

**Usage**

```
.validate_required(x, param_name)
```

**Arguments**

|                         |                        |
|-------------------------|------------------------|
| <code>x</code>          | Value to check         |
| <code>param_name</code> | Name for error message |

---

.validate\_research\_inputs      *Validate Research Inputs*

---

### Description

Validate Research Inputs

### Usage

```
.validate_research_inputs(  
    query,  
    schema,  
    output,  
    max_workers,  
    max_rounds,  
    budget,  
    stop_policy,  
    sources,  
    checkpoint_dir,  
    resume_from  
)
```

---

.validate\_run\_agent      *Validate run\_agent() Parameters*

---

### Description

Validate run\_agent() Parameters

### Usage

```
.validate_run_agent(prompt, agent, recursion_limit, verbose)
```

---

.validate\_run\_task      *Validate run\_task() Parameters*

---

### Description

Validate run\_task() Parameters

### Usage

```
.validate_run_task(prompt, output_format, agent, verbose)
```

---

|                          |   |
|--------------------------|---|
| .validate_run_task_batch | <i>Validate run_task_batch() Parameters</i> |
|--------------------------|---|

---

**Description**

Validate run\_task\_batch() Parameters

**Usage**

```
.validate_run_task_batch(  
    prompts,  
    output_format,  
    agent,  
    parallel,  
    workers,  
    progress  
)
```

---

|                    |                          |
|--------------------|--------------------------|
| .validate_s3_class | <i>Validate S3 Class</i> |
|--------------------|--------------------------|

---

**Description**

Validate S3 Class

**Usage**

```
.validate_s3_class(x, param_name, expected_class)
```

**Arguments**

- x                      Value to check
- param\_name            Name for error message
- expected\_class        Expected S3 class name

---

|                               |                                  |
|-------------------------------|----------------------------------|
| <code>.validate_string</code> | <i>Validate Non-Empty String</i> |
|-------------------------------|----------------------------------|

---

**Description**

Validate Non-Empty String

**Usage**

```
.validate_string(x, param_name, allow_empty = FALSE, allow_na = FALSE)
```

**Arguments**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>x</code>           | Value to check                       |
| <code>param_name</code>  | Name for error message               |
| <code>allow_empty</code> | Allow empty strings (default: FALSE) |
| <code>allow_na</code>    | Allow NA values (default: FALSE)     |

---

|                                      |  |
|--------------------------------------|--|
| <code>.validate_string_vector</code> | <i>Validate Character Vector (Non-Empty)</i> |
|--------------------------------------|--|

---

**Description**

Validate Character Vector (Non-Empty)

**Usage**

```
.validate_string_vector(x, param_name, min_length = 1L)
```

**Arguments**

|                         |                                      |
|-------------------------|--------------------------------------|
| <code>x</code>          | Value to check                       |
| <code>param_name</code> | Name for error message               |
| <code>min_length</code> | Minimum required length (default: 1) |

```
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_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)
```

**Arguments**

- python\_agent     The underlying Python agent object
- backend           LLM backend name (e.g., "openai", "groq")
- model            Model identifier
- config            Agent configuration list

**Value**

An object of class asa\_agent

---

|               |  |
|---------------|--|
| asa_enumerate | <i>Multi-Agent Research for Open-Ended Queries</i> |
|---------------|--|

---

**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,  
  schema = NULL,  
  output = c("data.frame", "csv", "json"),  
  max_workers = 4L,  
  max_rounds = 8L,  
  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),  
  pagination = TRUE,  
  progress = TRUE,  
  include_provenance = FALSE,  
  checkpoint = TRUE,  
  checkpoint_dir = tempdir(),  
  resume_from = NULL,
```

```

agent = NULL,
backend = "openai",
model = "gpt-4.1-mini",
conda_env = "asa_env",
verbose = TRUE
)

```

## Arguments

|                    |  |
|--------------------|--|
| query              | Character string describing the research goal. 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".   |
| max_workers        | Maximum number of parallel search workers (default: 4).  |
| max_rounds         | Maximum research iterations (default: 8).  |
| budget             | Named list with resource limits: <ul style="list-style-type: none"> <li>queries: Maximum search queries (default: 50)</li> <li>tokens: Maximum LLM tokens (default: 200000)</li> <li>time_sec: Maximum execution time in seconds (default: 300)</li> </ul>   |
| stop_policy        | Named list with stopping criteria: <ul style="list-style-type: none"> <li>target_items: Stop when this many items found (NULL = unknown)</li> <li>plateau_rounds: Stop after N rounds with no new items (default: 2)</li> <li>novelty_min: Minimum new items ratio per round (default: 0.05)</li> <li>novelty_window: Window size for novelty calculation (default: 20)</li> </ul> |
| sources            | Named list controlling which sources to use: <ul style="list-style-type: none"> <li>web: Use DuckDuckGo web search (default: TRUE)</li> <li>wikipedia: Use Wikipedia (default: TRUE)</li> <li>wikidata: Use Wikidata SPARQL for authoritative enumerations (default: TRUE)</li> </ul>  |
| 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 each round (default: TRUE).   |
| checkpoint_dir     | Directory for checkpoint files (default: tempdir()).   |
| resume_from        | Path to checkpoint file to resume from (default: NULL).  |
| 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", "openrouter".  |
| model              | Model identifier if creating new agent.  |
| conda_env          | Conda environment name (default: "asa_env").   |
| verbose            | Print status messages (default: TRUE).   |

## Details

The function uses a multi-agent architecture:

1. **Planner:** Decomposes query into facets and identifies authoritative sources
2. **Dispatcher:** Spawns parallel workers for each facet
3. **Workers:** Execute searches using DDG, Wikipedia, and Wikidata
4. **Extractor:** Normalizes results to match schema
5. **Deduper:** Removes duplicates using hash + fuzzy matching
6. **Stopper:** Evaluates stopping criteria (novelty, budget, saturation)

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 each round. If interrupted, use `resume_from` to continue from the last checkpoint:

```
result <- asa_enumerate(query, 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"
)

## 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_response | <i>Constructor for asa_response Objects</i> |
|--------------|---|

---

**Description**

Creates an S3 object representing an agent response.

**Usage**

```
asa_response(
  message,
  status_code,
  raw_response,
  trace,
  elapsed_time,
  fold_count,
  prompt
)
```

**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                |
| fold_count   | Number of memory folds performed         |
| prompt       | The original prompt                      |

**Value**

An object of class `asa_response`

---

|            |   |
|------------|---|
| asa_result | <i>Constructor for asa_result Objects</i> |
|------------|---|

---

**Description**

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

**Usage**

```
asa_result(prompt, message, parsed, raw_output, elapsed_time, status)
```

**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") |

**Value**

An object of class `asa_result`

---

|               |   |
|---------------|---|
| build_backend | <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 = "asa_env", conda = "auto", python_version = "3.13")
```

**Arguments**

|                |  |
|----------------|--|
| conda_env      | Name of the conda environment (default: "asa_env") |
| conda          | Path to conda executable (default: "auto")         |
| python_version | Python version to use (default: "3.13")            |

**Details**

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

- langchain\_groq, langchain\_community, langchain\_openai
- langgraph
- ddgs (DuckDuckGo search)
- selenium, primp (browser automation)
- beautifulsoup4, requests
- fake\_headers, httpx
- pysocks, socksio (proxy support)

**Value**

Invisibly returns NULL; called for side effects.

**Examples**

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

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

## End(Not run)
```

---

build\_prompt

*Build a Task Prompt from Template*


---

**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

*Check Python Environment Availability***Description**

Checks if the required Python environment and packages are available.

**Usage**

```
check_backend(conda_env = "asa_env")
```

**Arguments**

conda\_env      Name of the conda environment to check

**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 packages (if any)

**Examples**

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

## End(Not run)
```

---

|                  |                         |
|------------------|-------------------------|
| clean_whitespace | <i>Clean Whitespace</i> |
|------------------|-------------------------|

---

**Description**

Normalizes whitespace in a string by collapsing multiple spaces and trimming leading/trailing whitespace.

**Usage**

```
clean_whitespace(x)
```

**Arguments**

|   |                  |
|---|------------------|
| x | Character string |
|---|------------------|

**Value**

Cleaned string

---

|                  |   |
|------------------|---|
| configure_search | <i>Configure Python Search Parameters</i> |
|------------------|---|

---

**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 = "asa_env"  
)
```

**Arguments**

|             |  |
|-------------|--|
| max_results | Maximum number of search results to return (default: 10) |
| timeout     | HTTP request timeout in seconds (default: 15)            |
| 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: 0.5) |
| conda_env            | Name of the conda environment (default: "asa_env")           |

### 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 = "asa_env")
```

### Arguments

|           |   |
|-----------|---|
| level     | Log level: "DEBUG", "INFO", "WARNING" (default), "ERROR", or "CRITICAL" |
| conda_env | Name of the conda environment (default: "asa_env")                      |

**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)
```

---

decode\_html

*Decode HTML Entities*

---

**Description**

Converts HTML entities to their character equivalents.

**Usage**

```
decode_html(x)
```

**Arguments**

x                      Character string with HTML entities

**Value**

Decoded string

---

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 output string from agent invocation (the trace field from an asa_response object) |
|------------|---|

### 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 as a list (if present)
- search\_tiers: Character vector of unique search tiers used (e.g., "primp", "selenium", "ddgs", "requests")

### Examples

```
## Not run:
response <- run_agent("Who is the president of France?", agent)
extracted <- extract_agent_results(response$trace)
print(extracted$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)
```

**Arguments**

text                      Raw agent trace text

**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                      Raw agent trace text

**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)

**Arguments**

text                      Raw agent trace text

**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                      Raw agent trace text

**Value**

Character vector of Wikipedia snippets

**Examples**

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

## End(Not run)
```

---

|                 |                             |
|-----------------|-----------------------------|
| format_duration | <i>Format Time Duration</i> |
|-----------------|-----------------------------|

---

**Description**

Formats a numeric duration (in minutes) as a human-readable string.

**Usage**

```
format_duration(minutes)
```

**Arguments**

|         |                             |
|---------|-----------------------------|
| minutes | Numeric duration in minutes |
|---------|-----------------------------|

**Value**

Formatted string

---

|           |                              |
|-----------|------------------------------|
| 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")
```

### Arguments

|       |               |
|-------|---------------|
| proxy | Tor proxy URL |
|-------|---------------|

### Value

IP address string or NA on failure

### Examples

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

## 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(
  backend = "openai",
  model = "gpt-4.1-mini",
  conda_env = "asa_env",
  proxy = "socks5h://127.0.0.1:9050",
  use_memory_folding = TRUE,
  memory_threshold = 4L,
  memory_keep_recent = 2L,
  rate_limit = 0.2,
  timeout = 120L,
  verbose = TRUE
)
```

**Arguments**

|                    |   |
|--------------------|---|
| backend            | LLM backend to use. One of: "openai", "groq", "xai", "exo", "openrouter"                      |
| 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              | SOCKS5 proxy URL for Tor (default: "socks5h://127.0.0.1:9050"). Set to NULL to disable proxy. |
| use_memory_folding | Enable DeepAgent-style memory compression (default: TRUE)                                     |
| memory_threshold   | Number of messages before folding triggers (default: 4)                                       |
| memory_keep_recent | Number of recent messages to preserve after folding (default: 2)                              |
| rate_limit         | Requests per second for rate limiting (default: 0.2)  |
| timeout            | Request timeout in seconds (default: 120)   |
| verbose            | Print status messages (default: TRUE)   |

**Details**

The agent is created with two tools:

- Wikipedia: For looking up encyclopedic information
- DuckDuckGo Search: For web searches with a 4-tier fallback system (PRIMP -> Selenium -> DDGS library -> raw requests)

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
- OpenRouter: OPENROUTER\_API\_KEY

**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\\_agent](#), [run\\_task](#)

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)
```

---

 json\_escape

*Clean Text for JSON Output*


---

**Description**

Escapes special characters in text for safe inclusion in JSON strings.

**Usage**

```
json_escape(x)
```

**Arguments**

x                      Character string to escape

**Value**

Escaped string

---

 print.asa\_agent

*Print Method for asa\_agent Objects*


---

**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_enumerate_result
```

*Print Method for asa\_enumerate\_result Objects*

---

### 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_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

---

|        |                      |
|--------|----------------------|
| print2 | <i>Print Utility</i> |
|--------|----------------------|

---

**Description**

Wrapper around cat for consistent output formatting.

**Usage**

```
print2(...)
```

**Arguments**

|     |                         |
|-----|-------------------------|
| ... | Arguments passed to cat |
|-----|-------------------------|

---

|                 |                                       |
|-----------------|---------------------------------------|
| 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</i> |
|--------------------|---------------------------|

---

**Description**

Requests a new Tor circuit by restarting the Tor service.

**Usage**

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

**Arguments**

|        |   |
|--------|---|
| method | Method to restart: "brew" (macOS), "systemctl" (Linux), or "signal" |
| wait   | Seconds to wait for new circuit (default: 12)                       |

**Value**

Invisibly returns NULL

**Examples**

```
## Not run:
rotate_tor_circuit()

## End(Not run)
```

---

|           |   |
|-----------|---|
| run_agent | <i>Run the ASA Agent with a Custom Prompt</i> |
|-----------|---|

---

**Description**

Invokes the search agent with an arbitrary prompt, returning the full agent trace and response. This is the low-level function for running the agent; for structured task execution, use [run\\_task](#).

**Usage**

```
run_agent(prompt, agent = NULL, recursion_limit = NULL, verbose = FALSE)
```

**Arguments**

|                 |   |
|-----------------|---|
| prompt          | The prompt to send to the agent   |
| agent           | An asa_agent object from <a href="#">initialize_agent</a> , or NULL to use/create the default agent |
| recursion_limit | Maximum number of agent steps (default: 100 for memory folding, 20 otherwise)                       |
| verbose         | Print status messages (default: FALSE)  |

Value

- An object of class `asa_response` containing:
- `message`: The final response text
  - `status_code`: 200 for success, 100 for error
  - `raw_response`: The full Python response object
  - `trace`: Full text trace of agent execution
  - `elapsed_time`: Execution time in minutes
  - `fold_count`: Number of memory folds (if memory folding enabled)

See Also

[initialize\\_agent](#), [run\\_task](#)

Examples

```
## Not run:
# Run with a custom prompt
agent <- initialize_agent()
result <- run_agent(
  prompt = "Who was the 44th president of the United States?",
  agent = agent
)
print(result$message)

## End(Not run)
```

---

|                 |                                |
|-----------------|--------------------------------|
| run_agent_batch | <i>Run Agent in Batch Mode</i> |
|-----------------|--------------------------------|

---

Description

Runs the agent on multiple prompts, optionally in parallel.

Usage

```
run_agent_batch(
  prompts,
  agent = NULL,
  parallel = FALSE,
  workers = 4L,
  progress = TRUE
)
```

Arguments

|                       |  |
|-----------------------|--|
| <code>prompts</code>  | Character vector of prompts  |
| <code>agent</code>    | An <code>asa_agent</code> object                                     |
| <code>parallel</code> | Use parallel processing (requires <code>future.apply</code> package) |
| <code>workers</code>  | Number of parallel workers (default: 4)                              |
| <code>progress</code> | Show progress bar (default: TRUE)                                    |

**Value**

A list of `asa_response` objects

**Examples**

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

## End(Not run)
```

---

|          |   |
|----------|---|
| run_task | <i>Run a Structured Task with the Agent</i> |
|----------|---|

---

**Description**

Executes a research task using the AI search agent with a structured prompt and returns parsed results.

**Usage**

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

**Arguments**

|               |  |
|---------------|--|
| prompt        | The task prompt or question for the agent to research  |
| output_format | Expected output format. One of: "text" (raw response), "json" (parse as JSON), or a character vector of field names to extract |
| agent         | An <code>asa_agent</code> object from <a href="#">initialize_agent</a> , or NULL to use the currently initialized agent        |
| verbose       | Print progress messages (default: FALSE)   |

**Details**

This function provides a high-level 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.

**Value**

An object of class `asa_result` with components:

- `prompt`: The original prompt
- `message`: The agent's response text
- `parsed`: Parsed output (if `output_format` specified)
- `raw_output`: Full agent trace
- `elapsed_time`: Execution time in minutes
- `status`: "success" or "error"

**See Also**

[initialize\\_agent](#), [run\\_agent](#), [run\\_task\\_batch](#)

**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)

## End(Not run)
```

---

run\_task\_batch

*Run Multiple Tasks in Batch*


---

**Description**

Executes multiple research tasks, optionally in parallel.

**Usage**

```
run_task_batch(
  prompts,
  output_format = "text",
  agent = NULL,
  parallel = FALSE,
  workers = 4L,
  progress = TRUE
)
```

**Arguments**

|               |  |
|---------------|--|
| prompts       | Character vector of task prompts, or a data frame with a 'prompt' column |
| output_format | Expected output format (applies to all tasks)                            |
| agent         | An <code>asa_agent</code> object   |

|          |                            |
|----------|----------------------------|
| parallel | Use parallel processing    |
| workers  | Number of parallel workers |
| progress | Show progress messages     |

**Value**

A list of `asa_result` objects, or if `prompts` was a data frame, the data frame with result columns added

**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)

## End(Not run)
```

---

|                 |                        |
|-----------------|------------------------|
| safe_json_parse | <i>Safe JSON Parse</i> |
|-----------------|------------------------|

---

**Description**

Attempts to parse JSON, returning NULL on failure.

**Usage**

```
safe_json_parse(x)
```

**Arguments**

|   |             |
|---|-------------|
| x | JSON string |
|---|-------------|

**Value**

Parsed R object or NULL

---

|                   |   |
|-------------------|---|
| summary.asa_agent | <i>Summary Method for asa_agent Objects</i> |
|-------------------|---|

---

**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_enumerate_result | <i>Summary Method for asa_enumerate_result Objects</i> |
|------------------------------|--|

---

**Description**

Summary Method for asa\_enumerate\_result Objects

**Usage**

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

**Arguments**

|        |                                |
|--------|--------------------------------|
| object | An asa_enumerate_result object |
| ...    | Additional arguments (ignored) |

**Value**

Invisibly returns a 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

---

|                 |                        |
|-----------------|------------------------|
| truncate_string | <i>Truncate String</i> |
|-----------------|------------------------|

---

**Description**

Truncates a string to a maximum length, adding ellipsis if truncated.

**Usage**

```
truncate_string(x, max_length = 100, ellipsis = "...")
```

**Arguments**

|            |                                 |
|------------|---------------------------------|
| x          | Character string                |
| max_length | Maximum length                  |
| ellipsis   | String to append when truncated |

**Value**

Truncated string

---

|  |
|--|
| write_csv.asa_enumerate_result           |
| <i>Write asa_enumerate_result to CSV</i> |

---

**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

## \* internal

- [.build\\_trace, 4](#)
- [.close\\_http\\_clients, 5](#)
- [.create\\_agent, 5](#)
- [.create\\_http\\_clients, 6](#)
- [.create\\_llm, 6](#)
- [.create\\_research\\_config, 7](#)
- [.create\\_research\\_graph, 7](#)
- [.create\\_tools, 7](#)
- [.extract\\_fields, 8](#)
- [.extract\\_json\\_from\\_trace, 8](#)
- [.extract\\_json\\_object, 8](#)
- [.extract\\_response\\_text, 9](#)
- [.get\\_extdata\\_path, 9](#)
- [.get\\_local\\_ip, 9](#)
- [.get\\_python\\_path, 10](#)
- [.handle\\_response\\_issues, 10](#)
- [.import\\_python\\_packages, 10](#)
- [.import\\_research\\_modules, 10](#)
- [.invoke\\_memory\\_folding\\_agent, 11](#)
- [.invoke\\_standard\\_agent, 11](#)
- [.is\\_initialized, 11](#)
- [.normalize\\_schema, 11](#)
- [.parse\\_json\\_response, 12](#)
- [.process\\_research\\_results, 12](#)
- [.resume\\_research, 12](#)
- [.run\\_research, 12](#)
- [.run\\_research\\_with\\_progress, 13](#)
- [.save\\_checkpoint, 13](#)
- [.stop\\_validation, 13](#)
- [.validate\\_asa\\_agent, 14](#)
- [.validate\\_asa\\_response, 14](#)
- [.validate\\_asa\\_result, 14](#)
- [.validate\\_build\\_backend, 15](#)
- [.validate\\_build\\_prompt, 15](#)
- [.validate\\_choice, 15](#)
- [.validate\\_conda\\_env, 16](#)
- [.validate\\_configure\\_search, 16](#)
- [.validate\\_consistency, 17](#)
- [.validate\\_dataframe, 17](#)
- [.validate\\_initialize\\_agent, 18](#)
- [.validate\\_logical, 18](#)
- [.validate\\_positive, 19](#)
- [.validate\\_process\\_outputs, 19](#)
- [.validate\\_proxy\\_url, 19](#)
- [.validate\\_range, 20](#)
- [.validate\\_required, 20](#)
- [.validate\\_research\\_inputs, 21](#)
- [.validate\\_run\\_agent, 21](#)
- [.validate\\_run\\_task, 21](#)
- [.validate\\_run\\_task\\_batch, 22](#)
- [.validate\\_s3\\_class, 22](#)
- [.validate\\_string, 23](#)
- [.validate\\_string\\_vector, 23](#)
- [asa-package, 3](#)
- [clean\\_whitespace, 33](#)
- [decode\\_html, 35](#)
- [format\\_duration, 39](#)
- [json\\_escape, 43](#)
- [print2, 45](#)
- [safe\\_json\\_parse, 51](#)
- [truncate\\_string, 54](#)
- [.build\\_trace, 4](#)
- [.close\\_http\\_clients, 5](#)
- [.create\\_agent, 5](#)
- [.create\\_http\\_clients, 6](#)
- [.create\\_llm, 6](#)
- [.create\\_research\\_config, 7](#)
- [.create\\_research\\_graph, 7](#)
- [.create\\_tools, 7](#)
- [.extract\\_fields, 8](#)
- [.extract\\_json\\_from\\_trace, 8](#)
- [.extract\\_json\\_object, 8](#)
- [.extract\\_response\\_text, 9](#)
- [.get\\_extdata\\_path, 9](#)
- [.get\\_local\\_ip, 9](#)
- [.get\\_python\\_path, 10](#)
- [.handle\\_response\\_issues, 10](#)
- [.import\\_python\\_packages, 10](#)
- [.import\\_research\\_modules, 10](#)
- [.invoke\\_memory\\_folding\\_agent, 11](#)
- [.invoke\\_standard\\_agent, 11](#)
- [.is\\_initialized, 11](#)
- [.normalize\\_schema, 11](#)
- [.parse\\_json\\_response, 12](#)
- [.process\\_research\\_results, 12](#)

- .resume\_research, 12
- .run\_research, 12
- .run\_research\_with\_progress, 13
- .save\_checkpoint, 13
- .stop\_validation, 13
- .validate\_asa\_agent, 14
- .validate\_asa\_response, 14
- .validate\_asa\_result, 14
- .validate\_build\_backend, 15
- .validate\_build\_prompt, 15
- .validate\_choice, 15
- .validate\_conda\_env, 16
- .validate\_configure\_search, 16
- .validate\_consistency, 17
- .validate\_dataframe, 17
- .validate\_initialize\_agent, 18
- .validate\_logical, 18
- .validate\_positive, 19
- .validate\_process\_outputs, 19
- .validate\_proxy\_url, 19
- .validate\_range, 20
- .validate\_required, 20
- .validate\_research\_inputs, 21
- .validate\_run\_agent, 21
- .validate\_run\_task, 21
- .validate\_run\_task\_batch, 22
- .validate\_s3\_class, 22
- .validate\_string, 23
- .validate\_string\_vector, 23
  
- as.data.frame.asa\_enumerate\_result, 24
- as.data.frame.asa\_result, 24
- asa (asa-package), 3
- asa-package, 3
- asa\_agent, 25
- asa\_enumerate, 25
- asa\_enumerate\_result, 28
- asa\_response, 29
- asa\_result, 30
  
- build\_backend, 4, 30
- build\_prompt, 31
  
- check\_backend, 32
- clean\_whitespace, 33
- configure\_search, 33
- configure\_search\_logging, 34
  
- decode\_html, 35
  
- extract\_agent\_results, 36
- extract\_search\_snippets, 36
- extract\_search\_tiers, 37
  
- extract\_urls, 38
- extract\_wikipedia\_content, 38
  
- format\_duration, 39
  
- get\_agent, 39
- get\_tor\_ip, 40
  
- initialize\_agent, 4, 27, 40, 47–50
- is\_tor\_running, 42
  
- json\_escape, 43
  
- print.asa\_agent, 43
- print.asa\_enumerate\_result, 44
- print.asa\_response, 44
- print.asa\_result, 45
- print2, 45
- process\_outputs, 46
  
- reset\_agent, 46
- rotate\_tor\_circuit, 47
- run\_agent, 4, 42, 47, 50
- run\_agent\_batch, 48
- run\_task, 4, 27, 42, 47, 48, 49
- run\_task\_batch, 4, 50, 50
  
- safe\_json\_parse, 51
- summary.asa\_agent, 52
- summary.asa\_enumerate\_result, 52
- summary.asa\_response, 53
- summary.asa\_result, 53
  
- truncate\_string, 54
  
- write\_csv.asa\_enumerate\_result, 54