

A Baseline Assessment for the Agentic Web Technology Stack

VAGNER FIGUEREDO DE SANTANA, NIC.br, Brazil

DIOGO CORTIZ, NIC.br / PUC-SP, Brazil

The fast pace in which protocols, frameworks, and technologies are being proposed for LLM-based agents results in a complex scenario for anyone aiming to create state-of-the-art AI solutions. Without an assessment on how different protocols, frameworks, and libraries work together to compose an Agentic Web solution, developers and organizations face different challenges (e.g., skilling, governance, infrastructure) even before becoming part of the Agentic Web ecosystem. In this context, we propose a system for performing Agentic Web evaluations called MASTIF (Multi-Agent System TestIng Framework) and present a baseline assessment of Agentic Web technologies, including 3 LLMs, 3 protocols, and 4 agent implementation frameworks, against 300 *Mind2Web* benchmark tasks. Results suggest that selecting a task-specific technology stack may result in gains of task completion rate, efficiency score, and social value stability scores; and reductions in verbosity and time to complete tasks. With the presented results, we expect to present a baseline in terms of metrics and results for this complex and fast evolving scenario for those aiming at designing, assessing, and developing multi-agent systems.

CCS Concepts: • **Computing methodologies** → **Multi-agent systems**; **Cooperation and coordination**.

Additional Key Words and Phrases: Agentic Systems, Agentic Web, Multi-Agent Systems, AI-agents, LLM-agents, Agent Protocols, Agentic Protocol, Model Protocol

1 Introduction

In classical AI literature, an agent is “*anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators*” [50]. Agents that can act beyond their built-in knowledge are told to be autonomous [50]. In an effort to differentiate between software program and agents, Franklin and Graesser [24] proposed that “*An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.*” With recent advances in AI (Artificial Intelligence) and Large Language Models (LLMs), the goals envisioned with the idea of *perceiving* and *acting* were augmented by a natural language space [70]. The capabilities brought by LLMs applied to AI agents powered the now popular LLM-based agents. These agents may act on model of their environments, learn about their operation, or involve complex orchestration [57], being more useful for tasks that are harder, more realistic, and usually don’t have a single correct answer [30]. In the content that follows, we use *LLM-based agents* and *agents* interchangeably to align with current research in the area.

With the emergence of LLMs, the information previously available only on static Web pages is now embedded in LLMs themselves [68]. In addition, approaches such as Chain-of-Thought (CoT) [63] and *ReAct* [70] lead agents to perform intermediate reasoning steps, enabling planning and acting capabilities. With this transformation in place, the Web is transitioning from the ecosystem where only (human) users consume and create content/services to the *environment* in which agents are also part of this environment, *perceiving* and *acting* to perform high level, complex tasks, leading to the Agentic Web.

Authors’ Contact Information: Vagner Figueredo de Santana, NIC.br, São Paulo, Brazil, vsantana@nic.br; Diogo Cortiz, NIC.br / PUC-SP, São Paulo, Brazil, diogo@nic.br.



This work is licensed under a Creative Commons Attribution 4.0 International License.

Agentic Web can be defined as the ecosystem in which autonomous agents plan, coordinate, and execute goal-directed tasks, enabling continuous agent-agent interaction and value creation alongside human-Web interactions [68]. The term agentic was proposed as a way to cover the whole spectrum of use of agents rather than defining what an agent is or isn't [30]. Agent-agent communication here is considered to be the interaction that might not be rendered for human eyes, i.e., a new layer that involves optimized communication for agents in terms of parsing, planning, reasoning, and orchestration. Its core aspects include [68]:

- (1) Agents must function autonomously performing complex tasks independently,
- (2) Web resources need to be accessible to machines, and
- (3) Value must be exchanged between humans, systems, and agents.

Given that LLMs can use external APIs and tools in a self-supervised way [52], it is expected that the agent-agent interaction to increase, bringing challenges, opportunities, and risks, as this new layer is placed between the (human) end-user and the content/services available on the Web to be consumed/created by (digital) agents. This agent-agent interaction can also be referred to as **agent-as-user**, in which agents operate autonomously as users of the Web, executing tasks, navigating interfaces, and completing workflows [68]. Additional models of interaction include **agent-as-interface**, where agents enhance traditional user interfaces (e.g., browser), and **agent-with-physics**, where agents perceive, reason, and act in the physical world [68].

In many modern systems, autonomous agents do not operate alone but as **multi-agent systems** (MAS), coordinating, negotiating, or delegating tasks among a team of specialized agents. Some agents act as “orchestrators” that manage workflows, assign sub-tasks, or monitor progress across sub-agents. These agents combine perception, planning, memory, and coordination faculties to drive large-scale or hierarchical task completion [57]. MAS have also proven value for open-ended research tasks and for breadth-first queries, pursuing multiple directions simultaneously [9]. Results suggest that MAS with 4 agents outperform by 90% single agent for such tasks [9]. However, the use of agents has impacts in terms of token usage. In practice, single agents spent four times more tokens than LLM-powered chatbots and MAS spent up to fifteen times more tokens than LLM-powered chatbots [9]. This also highlights a key challenge, i.e., economic viability for MAS, requiring high value tasks to justify its use for the increased performance [9]. In Agentic Web, MAS potentially involve agents interacting with APIs, browsers, or data services to accomplish tasks beyond agent's internal reasoning.

Multiple protocols, frameworks, and tools are being proposed^{1,2}. However, this fragmentation makes it hard for developers and organizations to onboard agentic systems or simply to keep up with technologies for agents that work together, effectively. In this fast-paced complex evolving scenario [67, 68], this work aims to answer the following research questions (RQ) under the evaluated conditions for a baseline assessment:

- RQ1: What is the most effective (task completion rate) Agentic Web technology stack?
- RQ2: What is the most efficient (task completion rate, convergence time, and verbosity) Agentic Web technology stack?
- RQ3: What is the Agentic Web technology stack that is best suited for tool-calling on the Web?
- RQ4: What is the most stable Agentic Web technology stack in terms of social values embedded in their communication (e.g., prompts, reasoning, and output)?

¹<https://github.com/kaushikb11/awesome-llm-agents>

²<https://www.promptingguide.ai/research/llm-agents>

In this context, this work contributes with:

- (1) A system for performing empirical assessment of Agentic Web technology stack called MASTIF (Multi-Agent System TestIng Framework);
- (2) An empirical assessment baseline for the Agentic Web technology stack.

This paper is structured as follows: section 2 reviews the state-of-the-art on Agentic Web, section 3 formulates the problem, metrics, and use case considered in this paper, section 4 details the experiment performed, section 5 presents results obtained, section 6 articulates results towards the research questions guiding this research, and section 7 concludes.

2 Related Work

The recent wave of research on LLM agents has produced significant advances in reasoning, tool use, evaluation, and emerging interoperability standards. We group the most relevant works into the following main themes: types of agents, architecture, protocols, benchmarks, multi-agent system evaluations, and frameworks.

2.1 Types of Agents

Agents can be categorized along a spectrum of sophistication, memory, planning, and learning capabilities. The definitions presented next are useful for framing features and architectures discussed throughout this paper when assessing and comparing different technology stacks.

2.1.1 Simple reflex agents. These agents act purely on predefined condition–action rules, without memory or internal modeling, i.e., they map current percepts directly to actions. While suitable for very constrained or predictable domains, they cannot plan ahead or adapt based on previous experience [50, 57].

2.1.2 Model-based reflex agents. These agents have an internal representation (model) of their environment and update it over time, allowing better reactivity than reflex-only systems by considering inferred state changes. However, they are still fundamentally reactive and lack explicit goal-driven planning or long-term foresight [50, 57].

2.1.3 Goal-based agents. These agents explicitly reason about future states to choose sequences of actions that achieve specified goals. Planning is central to their decision-making, which allows them to decompose tasks and make multi-step decisions toward achieving objectives—even when the path is not obvious from immediate observations [50, 57]. *When no learning is involved, LLM-based agents often fall within this category.*

2.1.4 Utility-based agents. Beyond simply reaching a goal, utility-based agents evaluate and compare potential outcomes using a utility function, seeking not just any solution but the *most optimal* one according to specified criteria (e.g., speed, cost, accuracy, user satisfaction). This allows richer, more nuanced decision-making in tradeoff-laden environments [50, 57].

2.1.5 Learning agents. Learning agents refine their behaviors over time through feedback, experience, or training. They can utilize supervised, reinforcement, or unsupervised learning to update their internal models, policies, or plans—thereby improving performance in dynamic, uncertain, or previously unseen scenarios. *With learning in place, this category overlaps with LLM-based agents or other adaptive models [57].*

2.2 Architecture

2.2.1 AGNTCY. AGNTCY is an open source project aiming at structuring the Internet of Agents (IoA) by recommending a framework including³: (1) agent discovery, using Open Agent Schema Framework (OASF) to allow agents to discover and understand the capabilities of others, (2) agent identity, providing verifiable identity and access control across organizational boundaries, (3) agent messaging, supporting communications via SLIM (Secure Low-latency Interactive Messaging), (4) agent observability, providing end-to-end provenance in MAS, (5) protocol integration, supporting agent and model protocols.

2.2.2 Networked AI Agents in Decentralized Architecture (NANDA). Project NANDA proposes an infrastructure for an open Agentic Web by supporting how AI agents can discover each other, verify capabilities, and coordinate tasks without creating bottlenecks or security vulnerabilities. The project focuses on the technical infrastructure and the governance frameworks⁴.

2.3 Protocols

In this section, we present protocols for agentic systems. Given the focus on the Agentic Web, the scope of this work will cover general-purpose protocols. **General-purpose protocols** focus on supporting multiple agents and contexts via a unified interface, while **domain-specific protocols** focus on optimizing for specific use cases and contexts [67]. Moreover, it is important to contrast agent protocols from context protocols. **Inter-agent protocols** are referred as being the ones allowing components to initiate a task autonomously, while **context-oriented protocols** are the ones that support connection of tools and models [67].

2.3.1 Model Context Protocol (MCP). MCP is an open transport-agnostic context protocol that enables developers to build secure, two-way connections between their data sources and LLM-powered tools. The architecture allows developers to expose their data through MCP servers or build AI applications (MCP clients) that connect to these servers using JSON-RPC for communication [8]. Considering this aspect related to connection with tools, MCP has pre-built MCP servers for Google Drive, GitHub, Slack, databases, among others. Protocol main components: *Client, Server, Tools, Resources, Prompts, and Sampling*.

MCP is also promoting the development of new technologies for the Web to improve how agents perform tasks on the Web. An example is WebMCP, an open source JavaScript library that helps on how websites expose available tools in a MCP-integrated way. WebMCP provides a standardized way to register tools, prompts and resources so that agents know exactly how to interact with a website's features, improving reliability and task performance on the Agentic Web [1, 16]. WebMCP API methods: *registerTool, registerPrompt, and registerResource*.

2.3.2 Agent-to-Agent Protocol (A2A). A2A is an open client-server protocol that enables AI agents to communicate and collaborate across different platforms and frameworks, leveraging existing Web infrastructure. It's designed to maximize the benefits of agentic AI by enabling true multi-agent scenarios using JSON-RPC for communication. Key Features include interoperability (language- and framework-agnostic), enterprise-grade security, and flexibility [26]. Protocol main components: *Agent Card, Skill, Task, Message, and Artifact*.

2.3.3 Agent Communication Protocol (ACP). ACP is an open client-server protocol for agent interoperability aiming to support different frameworks and infrastructures. It considers agent-agent communication through a standardized RESTful API that supports different modalities,

³<https://agntcy.org/>

⁴<https://nanda.media.mit.edu/>

(a)synchronous communication, streaming interactions, stateful/less operations, agent discovery, and long tasks. Given that it is message-based, the protocol is agnostic to internal implementations [47]. Protocol main components: *Agent Detail, Discovery Mechanisms, Task Request, Message Structure, and Artifacts*.

2.3.4 Agent Network Protocol (ANP). ANP is an open-source peer-to-peer communication protocol for agents, defining how agents connect with each other, building an open, secure, and efficient collaboration network for agents ⁵. It is a standardized cross-domain protocol based on processes in place. Protocol main components: *Agent Identity, Host, Client, Server, and Resource*.

2.3.5 Agora Protocol. Agora is an open protocol that provides decentralized communication primitives designed for large-scale LLM agent ecosystems. It aims at supporting communication among heterogeneous, decentralized agents [5]. It translates natural language into structured protocols, serving as an intermediary layer that maps user’s task to the specific protocols required by agents. [67]

2.3.6 Agent Payments Protocol (AP2). AP2 is an open protocol aiming at compliant agent-led payments across platforms. The protocol can be used as an extension of the A2A and MCP. It involves two main components for transactions *Intent Mandate* and *Cart Mandate*, allowing the comparison of the initial human-led task (intent) and the product/server purchased by the agent (cart) ⁶.

Finally, instead of a specific stack of protocols, Ehtesham et al. [22] recommend a different approach, i.e., an adoption roadmap for protocols, starting with MCP for tool access, then ACP for multimodal messaging, then A2A for collaborative task execution, and, finally, extending to ANP for decentralized agent marketplaces. The rationale proposed by the authors is that “*no single protocol suffices across all contexts*” [22].

2.4 Agentic Frameworks

Building MAS requires frameworks to manage communication and orchestration among specialized roles. Given that Python is the most popular language⁷, our work will focus on frameworks in Python. Appendix A (Table 7) presents MAS frameworks with active open-source projects. From the list of frameworks studied, the LLM-agnostic popular frameworks with higher level of compatibility with built-in tools, allowing exploring different workflow paradigms, are: LangChain [31], CrewAI [17], LangGraph [32], and Semantic Kernel [54]

2.5 Benchmarks for Web Tasks

To systematically evaluate agents, several benchmarks and environments have been proposed. Appendix B (Table 8) presents a list of benchmarks used in assessments involving agentic environments. In the context of Agentic Web for agent-as-user, benchmarks that align with tasks and tools available at the Web were proposed in works such as Mind2Web [21], WebArena [72], and WebShop [69], detailed next.

WebArena provides a benchmark that aims at evaluating tasks that humans routinely perform on the Internet, encompassing the following domains: online shopping, discussion forums, collaborative development, and business content management. The content populating these self-hosted websites was extracted from their real-world counterparts. It covers 812 long-horizon web-based tasks, described as a high-level natural language intent [72].

⁵<https://github.com/agent-network-protocol/AgentNetworkProtocol>

⁶<https://github.com/google-agentic-commerce/AP2>

⁷<https://spectrum.ieee.org/top-programming-languages-2025>

WebShop provides a benchmark comprising simulated e-commerce website environment with 1.18 million real-world products scraped from Amazon.com and over 12,000 crowd-sourced text instructions (prompts) for agents to find a given target product. It aims at providing challenges for natural language processing for agents, including compositional instructions query formulation, and acting on noisy text [69].

Mind2Web provides a benchmark that aims at supporting the development of generalist agents for the Web that follow language instructions to complete complex tasks on any website. It was built to encompass three main aspects (1) diverse domains, websites and tasks, (2) use real-world websites, and (3) broad spectrum of user interaction patterns. Mind2Web encompasses over 2,000 tasks, from 137 websites, spanning 31 different application domains such as travel, information, service, shopping, and entertainment [21].

2.6 MAS Evaluations

Beyond benchmarks, context protocols and client-server communication has also been assessed in the literature. MCP Interviewer [43] was proposed as a tool to evaluate how MCP servers behave in multiple scenarios. Results suggest that MCP servers [23, 43]: (1) do not know which clients or models they are working with, (2) differ in terms of handling larger contexts, (3) respond differently to common prompting patterns, and (4) face name collision in terms of tools available (755 collisions under the term “search”).

In MAS, prompts tailored to agents, instilling good heuristics rather than rigid rules, results in better outcomes [9]. Anthropic provides the following guidelines for prompting for MAS [9]: (1) think like your agents, (2) teach the orchestrator how to delegate, (3) scale effort to query complexity, (4) tool design and selection are critical, (5) let agents improve themselves, (6) start wide, then narrow down, (7) guide the thinking process, and (8) parallel tool calling transforms speed and performance. The following extracts from a research MAS showcase these guidelines applied to prompting for agents ⁸:

- **Lead researcher agent (Orchestrator):** “[...] As the lead research agent, your primary role is to coordinate, guide, and synthesize - NOT to conduct primary research yourself. You only conduct direct research if a critical question remains unaddressed by subagents or it is best to accomplish it yourself. Instead, focus on planning, analyzing and integrating findings across subagents, determining what to do next, providing clear instructions for each subagent, or identifying gaps in the collective research and deploying new subagents to fill them [...]”
- **Research agent (Subagent):** “[...] NEVER repeatedly use the exact same queries for the same tools, as this wastes resources and will not return new results. Follow this process well to complete the task. Make sure to follow the description and investigate the best sources” [...] “when you are no longer finding new relevant information and results are not getting better, STOP using tools and instead compose your final report [...]”
- **Citation agent (Subagent):** “[...] ONLY add citations where the source documents directly support claims in the text [...]”

When performing evaluations against MAS, LLM-as-a-judge has shown to be an effective and scalable solution when the eval test cases have a clear answer [9]. When evaluating MAS, long conversations management is also an increasing challenge as some agents engage in conversations spanning hundreds of turns, cases in which compression and memory mechanisms are required such as summarization of completed work, storage of essential information in external memory before proceeding to new tasks, and spawning of subagents with clean contexts [9].

⁸<https://github.com/anthropics/claude-cookbooks/tree/main/patterns/agents/prompts>

3 Problem Definition

The formalization presented next aims at connecting the technology stack for agents with research questions posed in section 1 in a verifiable way. The metrics presented next allow for ranking technology stacks for agents and, hence, to identify promising directions in terms of technology for the Agentic Web.

First, let $\mathcal{A} = \{a_1, \dots, a_n\}$ denote a set of agents. Each agent instance a runs in an environment that combines a set of technologies defined in this study as a technology stack (TS)

$$TS = (LLM, AP, IF),$$

where LLM is the model agents are using, AP is the agent protocol, and IF the implementation framework.

A prompt P is a tuple $P = (T, C)$ where T is a high-level instruction (task) and C is a given initial context, which may be empty for initial prompts or even poorly defined prompts.

An orchestrator (or planner agent) a_{orch} may decompose T into subtasks $\{t_1, \dots, t_k\}$ and dispatch prompts $p_i = (t_i, c_i)$ to subagents a_i . The orchestrator collects subagent outputs o_{a_i} and returns a final answer o_{orch} .

The action space for an agent includes an LLM-powered language space $\mathcal{L}$ and output actions drawn from (a subset of) $\mathcal{L}$. Each agent a produces a reasoning trace a_t ; hence

$$\begin{aligned} a_{i,t1} &= TS(P(t_1, C)) \\ a_{i,t2} &= TS(P(t_1, a_{i,t1})) \\ a_{i,t3} &= TS(P(t_1, a_{i,t2})) \\ &\dots \\ a_{i,tm} &= TS(P(t_1, a_{i,t(m-1)})) \\ o_{ai} &= TS(P(t_1, a_{i,tm})) \end{aligned}$$

where a_t is the reasoning trace and o_a is the output (which may be used as a context).

3.1 Task Completion

Let $\Phi(T, C, O)$ be a task-specific predicate indicating whether output O satisfies the instructions in task T . As recommended in [9, 30], we consider Φ to be realized either by:

- (1) An LLM-as-a-judge with a rubric informing success/failure criteria, or
- (2) Human assessment with an success/failure criteria.

Given that the output for a given task is given by

$$o_{orch} = TS(P(T, \{o_{a1}, \dots, o_{an}\})),$$

we say a task instance (T, C) is *completed* under MAS technology stack TS if

$$\Lambda_{TS}(T, C) = \mathbb{I}\{\Phi(T, C, o_{orch}) = 1\},$$

where $\mathbb{I}\{\cdot\}$ is the indicator function.

Consider a test set $\mathcal{D} = \{(T_j, C_j)\}_{j=1}^{|\mathcal{D}|}$, we define the Task Completion Rate (TCR) as

$$TCR_{TS}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{P \in \mathcal{D}} \Lambda_{TS}(T, C) \quad (1)$$

3.2 Efficiency

Efficiency can be seen as the metric that combines high effectiveness with low use of resources. Hence, in the context of this work, we propose assessing efficiency as a multi-dimensional efficiency profile composed of: task completion, convergence time (or the time a MAS takes to answer a task), and verbosity (or tokens spent by a MAS to answer a task).

Convergence time (CT) is then defined as the time taken by agents to converge and provide an answer for a given prompt. For a given task instance with prompt $P = (T, C)$, let $T(\cdot)$ denote the time elapsed from the moment the orchestrator receives P until the system it produces a final output o_{orch} .

$$CT_{TS}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{P \in \mathcal{D}} T\{\Lambda_{TS}(T, C)\}. \quad (2)$$

We now focus the efficiency formulation specifically for each TS. The goal is to quantify how choices in TS affect reasoning traces and the tokens generated. In the context of GenAI, cost in terms of input and output tokens is an important metric for comparing solutions [30]. Hence, technology stack efficiency (E) takes into account internal reasoning and tokens used. Hence, verbosity (V) is defined in terms of average tokens spent per task:

$$V_{TS}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{P \in \mathcal{D}} \sum_{a \in \mathcal{A}} \|a_{it}\| + \|o_{ai}\| \quad (3)$$

Where $\|a_{it}\|$ represents the total number of tokens in the reasoning trace for agent a_i and $\|o_{ai}\|$ the total number of tokens of the output generated by an agent a_i . Agents may generate output o_{ai} with no reasoning steps, i.e., $\|a_{it}\| = 0$.

With these components in hand, efficiency can be seen as a multi-dimensional efficiency profile considering task completion, convergence time, and verbosity of agents, represented respectively by TCR_{TS} , CT_{TS} , and V_{TS} .

$$E_{TS}(\mathcal{D}) = (TCR_{TS}(\mathcal{D}), CT_{TS}(\mathcal{D}), V_{TS}(\mathcal{D})) \quad (4)$$

3.3 Tool Use

Given that agents have freedom to select tools given a set of provided tools, we need to assess how they perform under this condition. Hence, let a_{i,t_m} denote the reasoning trace of agent a_i at task t_m , where:

$$a_{i,t_m} = TS(P(t, a_{i,t_{m-1}})).$$

A reasoning trace a_{i,t_m} may trigger a tool call. Let $\mathcal{U}_{TS}$ be the set of all tool calls induced by agents performing tasks in $\mathcal{D}$. Each tool call $u \in \mathcal{U}_{TS}$ is associated with:

- A tool identifier $\theta(u)$, and
- An execution outcome $\delta(u) \in \{0, 1\}$.

Let $\mathcal{T}$ be the set of available tools. We define the tool use success indicator:

$$tsi(u) = \begin{cases} 1 & \text{if } \theta(u) \in \mathcal{T} \text{ and execution succeeds} \\ 0 & \text{otherwise (e.g., unsuccessful, hallucination, etc.)} \end{cases}$$

Hence, we define tool use success rate (TUSR) as the success rate over all tool calls:

$$TUSR_{TS}(\mathcal{D}) = \frac{1}{|\mathcal{U}_{TS}|} \sum_{u \in \mathcal{U}_{TS}} tsi(u). \quad (5)$$

3.4 Social Values

Let $SV = \{sv_1, sv_2, \dots, sv_x\}$ denote a set of social values, e.g., trust, compliance, integrity, and safety. Each prompt P consists of sentences $s \in P$, and a classifier assigns a $sv_i \in SV$ to each s . For an agent a_i , define its set of used social values (SV) for a given prompt as

$$SV_i(P) = \bigcup_{s \in P} \kappa(s), \quad (6)$$

Where P is the prompts handled by a_i and κ a classifier to retrieve a social value per sentence. We now formulate the value stability from prompt, internal reasoning steps, up to final output. Considering agents consistency, internal value stability (IVS) can be measured as a Pairwise Jaccard-based formula:

$$x_0 = P, \quad x_j = a_i t_j \quad (1 \leq j \leq m), \quad x_{m+1} = o_{a_i}, \quad S_j = SV(x_j)$$

$$IVS(a_i) = \frac{2}{(m+1)(m+2)} \sum_{j=0}^m \sum_{k=j+1}^{m+1} \frac{|S_j \cap S_k|}{|S_j \cup S_k|} \quad (7)$$

Where $a_{t1} \dots a_{tm}$ represent internal reasoning steps for an agent. P represents the initial prompt and o_{a_i} the final output provided by the orchestrator. Hence, internal value stability is used to compute how social values change through the process for a single task. Moreover, the social value stability (SVS) score for a technology stack TS is defined as:

$$SVS_{TS}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{P \in \mathcal{D}} IVS(a_i) \quad (8)$$

Where SVS is a score of social value stability for a technology stack through the test set $\mathcal{D}$.

4 Experiment Setup

In this section we detail the materials considered in this study, including models, protocols, benchmark, testing framework (MASTIF), and tools provided to agents. Then, we detail the procedure and result analysis, based on the problem definition proposed in Section 3.

4.1 Materials

4.1.1 Protocols. The rationale on considering MCP accompanied by agent protocols resides in the fact that the latter carry structured JSON and artifacts while the former will be responsible for routing tool calls through. Hence, MCP focuses on standardizing and how AI models and agents interact, in a structured way, with tools, APIs, data sources and any other external resource, while agent protocols aim on standardizing how independent agents interact with each other to discover, negotiate, delegate, share tasks, exchange contexts, or deliver results [2]. Given this context and the announcement that ACP is being merged with A2A [12], the selected protocols for this study are MCP and A2A (Table 1). Protocols were compared in terms of overhead applied to the communication among agents. The standard condition will be performed in terms of messages without protocols' directives in exchanged messages.

4.1.2 Frameworks. Considering the list of frameworks from the literature (Appendix A, Table 7), the LLM-agnostic popular frameworks with higher level of compatibility and easy integration with the protocols studied are: LangGraph [32], LangChain [31], CrewAI [17], and Semantic Kernel [54] (Table 1).

4.1.3 Benchmark. In terms of benchmark, considering the ones present in the literature (Appendix B, Table 8), the one that provides support for assessing generalist agents for the Web for real-world websites, considering multiple domains, is the Mind2Web [21]. The rationale was to consider a benchmark for agents to perform tasks involving e-commerce, travel planning, and enterprise knowledge assistants [68]. In this experiment, a stratified sample of 300 tasks was considered to encompass different websites and domains.

Examples of high-level tasks present in the Mind2Web [21] include:

- “Favorite a shared room in Belo Horizonte costing up to \$100 to stay in a weekend in May.”
- “Find the results of the most recent NFL games.”
- “Add a set of sonoma bath towels to the cart and apply a coupon code for free shipping.”
- “Find hotel deals in Las Vegas for four adults starting on May 17 and ending on May 20, and if deal is not available, set an alert for the same.”
- “Browse certified fresh sci-fi films streaming at home.”

4.1.4 Large language Models. The following models hosted on Hugging Face were selected based on capabilities for tool use and agent reasoning [6, 19, 29, 59]:

- Mistral-7B-Instruct-v0.1: Baseline for instruction following and efficiency.
- Qwen2.5-7B: Evaluating tool-use accuracy and structured output.
- Llama-4-Scout-17B-16E-Instruct: Long-context memory and multimodal reasoning.

Judge model selected is gpt-4o-mini in a way to consider a different family model to judge MAS responses. The rationale for this decision aims at preventing *preference leakage* as models from the same family tend to prefer outputs similar to their own distributions given shared LLM baselines and benchmarks [35], while controlling for costs [30] by selecting a cost-benefit model [45]. Table 1 summarizes the technology stack considered in this paper. Appendix C details judge model rubric used to assess MAS answers.

The rationale for selecting these models and judge was to balance cost and tool use in a MAS [30].

4.1.5 MAS Test Framework. In order to test different GenAI models and agentic frameworks, a testing framework was developed, MASTIF (Multi-Agent System TestIng Framework). It is open source and available on GitHub⁹. The goal of MASTIF is to support the specification of multiple tasks, a prompt template, connection with different inference providers (i.e., HuggingFace, OpenAI, Anthropic, and Ollama), and testing of multiple models and implementation frameworks. The testing framework logs latency, token usage, intermediate reasoning steps, and other metrics to enable the assessment of MAS on the Web. Figure 1 depicts MASTIF’s main components.

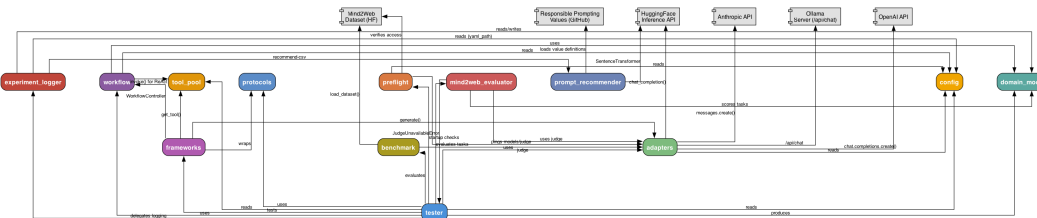


Fig. 1. Overview of Multi-Agent System TestIng Framework (MASTIF).

MASTIF offers the following set of tools to be used by agents. The tools include 13 built-in tools compatible with the frameworks selected and two custom tools (i.e., *web_interaction* and

⁹<https://github.com/cewebbr/mastif>

keyboard_interaction) that support interactivity with the Web, as needed for Mind2web tasks. The provided tools involve core & utility tools, knowledge tools, computation & parsing tools, content extraction tools, and interaction tools, as follows:

- *web_search*: DuckDuckGo web search (via duckduckgo search).
- *web_browser*: Playwright headless browser navigation.
- *wikipedia*: Wikipedia lookup (via wikipedia).
- *arxiv*: Academic paper search (via arxiv).
- *python_repl*: Sandboxed Python code execution (via Restricted Python).
- *requests_get*: Simple HTTP GET requests (via requests)
- *beautifulsoup_scraper*: HTML structure extraction (via beautifulsoup4).
- *pdf_reader*: PDF text extraction (via pypdf).
- *datetime*: Current date and time (stdlib).
- *json_parser*: JSON string parsing and querying (stdlib).
- *youtube_transcript*: YouTube video transcript retrieval (via youtube-transcript-api).
- *sympy*: Symbolic math and logic evaluation (via sympy).
- *web_interaction*: Interactive browser tool for multi-step web interactions (via Playwright).
- *keyboard_interaction*: Keyboard-driven browser interaction optimized for accessibility-style tasks (via Playwright).

Following good practices for describing tools to agents [7], the following tool descriptions were provided to agents to promote tool usage:

- *web_search*: Use this tool when you need to find information that is not already known or may be outdated. This includes current events, recent facts, or unknown topics. Input should be a clear search query. Do NOT use this for simple reasoning or when the answer is already known.
- *web_browser*: Use this tool to open and read the content of a specific webpage when you already have a URL. Useful after performing a search. Do NOT use this for searching—use *web_search* first.
- *requests_get*: Use this to retrieve raw content from a specific URL (API, webpage, or endpoint). Prefer this for structured or programmatic access. Do NOT use for general browsing or search.
- *beautifulsoup_scraper*: Use this to extract structured information from HTML content of a webpage. Typically used after fetching a page via *web_browser* or *requests_get*. Do NOT use if you only need raw text.
- *datetime*: Use this when you need the current date or time. Do NOT guess or fabricate current timestamps.
- *wikipedia*: Use this for general knowledge lookups on well-known topics, people, or concepts. Faster and more reliable than *web_search* for encyclopedic content. Do NOT use for recent or niche information.
- *arxiv*: Use this to find academic papers in technical or scientific domains. Input should be a research topic or keywords. Do NOT use for general knowledge or non-academic queries.
- *pubmed*: Use this for biomedical or clinical research queries. Input should be medical terms or research questions. Do NOT use for general health advice or non-scientific queries.
- *python_repl*: Use this for calculations, data processing, or executing code. Required for precise arithmetic, transformations, or simulations. Do NOT use for simple math that can be done mentally.
- *sympy*: Use this for symbolic mathematics such as algebra, equation solving, or simplification. Prefer this over *python_repl* for symbolic operations.

- *json_parser*: Use this to parse, extract, or query structured JSON data. Input should be a JSON string and query instructions. Do NOT use for plain text.
- *pdf_reader*: Use this to extract text from PDF files or PDF URLs. Required when the content is not directly readable as text. Do NOT use for HTML pages.
- *youtube_transcript*: Use this to retrieve the transcript of a YouTube video. Input should be a video URL. Do NOT use if you only need metadata or summaries.
- *web_interaction*: Use this to interact with webpages in multi-step tasks. Includes clicking elements, filling forms, navigating pages, and extracting results. Required for tasks involving user workflows on websites. Do NOT use for simple retrieval—use *web_search* or *web_browser* instead.
- *keyboard_interaction*: Use this for keyboard-based navigation of web interfaces. Includes tabbing between elements, pressing enter, typing text, and triggering shortcuts. Prefer this when interaction must simulate accessibility or non-mouse workflows. Do NOT use for static content extraction.

4.1.6 Social Value Classifier. The social value classifier considered in this assessment will be based on the work from Santana et al. [51], reusing the social value classification based on similarity with a corpus of human-curated prompts and social values, powered by light weight the sentence transformers such as *all-minilm-l6-v2*. The system retrieves a set of social value based on a given text input. The classifier thus retrieves a class (social value) for each sentence in the content provided, be it a task, prompt, agent reasoning step, or final answer. The recommendation system is open source and can be tested via its cookbook notebooks, available on Github¹⁰. This approach will be used to assess Equation 8.

4.2 Experiment Design and Procedure

Based on the selected benchmark and testing framework, the following set of models, frameworks, protocols, and tasks will be considered (Table 1).

$$\text{Tests} = |\text{models}| \times |\text{frameworks}| \times |\text{protocols}| \times |\text{tasks}| \quad (9)$$

Hence, based on Equation 9 and stratified sampling [25] of 300 out of 2,350 Mind2Web tasks, a total of $3 \times 4 \times 3 \times 300 = 10,800$ task executions will be evaluated. Instead of simple random sampling, stratification was employed to allow selecting tasks from different domains and tasks of different complexities [25]. Additional parameters considered in the experiment include:

- *temperature*: 0.7, aiming at diverse search interpretation; lower temperature tends to fix behavior on how agents use tools and higher temperature tends to allow different behaviors in terms of how to perform a task, tool use, including tool call hallucinations.
- *max_tokens*: 1024, to support cumulative {research_results}.
- *max_steps*: 3, to allow multiple agent loops if needed.
- *max_tool_ounds*: 3, to allow agents to use one or more tools multiple times.
- *timeout*: 100 seconds, increased timeout to handle automation restrictions placed by some search engines.
- *sampling_seed*: 20260504.

Beyond these parameters, MASTIF also supports defining the agentic workflow (linear and hierarchical), including nodes with and without loops, each node with its configurable prompt. Hence, considering the format of tasks in Mind2Web [21], the workflow used in this experiment considers 4 nodes in the following way:

¹⁰<https://github.com/IBM/responsible-prompting-api/tree/main/cookbook>

Table 1. Experiment setup overview.

Category	$TS_{model,protocol,framework}$
Models	• Mistral-7B-Instruct-v0.1 [29] • Qwen2.5-7B [59] • Llama-4-Scout-17B-16E-Instruct [6]
Judge model	• GPT-4o-mini [45]
Protocols	• Standard call • A2A [2] • MCP [8]
Frameworks	• LangGraph [32] • LangChain [31] • CrewAI [17] • Semantic Kernel [54]
Benchmark	• Mind2Web [21]

- (1) Navigating;
- (2) Observing the webpage at hand (loop node);
- (3) Interaction with the webpage at hand (loop node), and;
- (4) Final reporting.

Open models and tasks used HuggingFace API while judge model used OpenAI API. All the specifications for this experiments are detailed in the YAML¹¹ file ingested by the testing framework, prompts¹² considered in the workflow, and resulting logs¹³ are all available on GitHub.

4.3 Result Analysis

Result analysis will aim at comparing different Agentic Web technology stacks (TS) in terms of the following metrics, detailed in Section 3:

- TCR_{TS} : task completion rate (Equation 1),
- CT_{TS} : convergence time (Equation 2),
- V_{TS} : verbosity (Equation 3),
- E_{TS} : efficiency (Equation 4) plus a Pareto frontier analysis,
- $TUSR_{TS}$ tool use (Equation 5), and
- SVS_{TS} : social values stability (Equation 8).

Shapiro-Wilk will be used to test for normal distribution and then, depending on the result, ANOVA or Kruskal-Wallis plus Dunn’s Post-hoc tests for assessing significant differences considering models, protocols, and frameworks

¹¹<https://github.com/cewebbr/mastif/tree/main/experiments/tweb2026.yaml>

¹²<https://github.com/cewebbr/mastif/tree/main/prompts/mind2web/>

¹³<https://github.com/cewebbr/mastif/tree/main/logs/>

5 Results

In this section we present results for the metrics defined in Section 3. Shapiro-Wilk statistic was computed for all metrics detailed next and results suggest that they do not follow a normal distribution ($p < 0.001$). Hence, Kruskal-Wallis test was employed to assess significant differences considering models, protocols, and frameworks. Moreover, Dunn's Post-hoc test was used to identify pairwise significant differences ($\alpha = 0.05$)

5.1 Task Completion Rate

Task completion rate was computed according to Equation 1 and processed by the judge model according to the rubric detailed in Appendix C.

Kruskal-Wallis test for task completion rate suggests that there are significant differences for model (H-statistic = 1203.263, $p - value < 0.001$), protocol (H-statistic = 428.770, $p - value < 0.001$), and framework (H-statistic = 6039.33, $p - value < 0.001$).

Dunn's Post-hoc suggests significant pairwise differences for model when assessing task completing rates as follows:

- Llama-4-Scout-17B-16E vs. Mistral-7B ($p - value < 0.001$) and
- Llama-4-Scout-17B-16E vs. Qwen2.5-7B ($p - value < 0.001$).

Dunn's Post-hoc suggests significant pairwise differences for protocol when assessing task completing rates as follows:

- A2A vs. MCP ($p - value < 0.001$),
- A2A vs. standard ($p - value < 0.001$), and
- MCP vs. standard ($p - value < 0.001$).

Dunn's Post-hoc suggests significant pairwise differences for framework when assessing task completing rates as follows:

- CrewAI vs. SemanticKernel ($p - value < 0.001$),
- LangChain vs. SemanticKernel ($p - value < 0.001$), and
- LangGraph vs. SemanticKernel: $p - value < 0.001$).

Figure 2 shows a violin plot grouping task completion scores by model, protocol, and framework.

Table 2 shows mean scores obtained in technology stacks studied. Moreover, given that we also required judge model to describe its rationale and assessment of task understanding and how reasoning adheres to the task, we report additional scores retrieved by the model in terms of task understanding (TU_{TS}) and adherence (TA_{TS}), all according to the judge's provided rubric (Appendix C).

Hence, results suggest, under the test conditions, that the best-performing technology stack aiming at task completion score should consider: **Qwen2.5-7B** (mean rank=6095.99), **standard** (mean rank=5898.66), and **SemanticKernel** (mean rank=6039.33).

Table 2. Average task understanding (TU_{TS}), adherence (TA_{TS}), and task completion (TCR_{TS}) for different technology stacks, as provided by judge model given rubric detailed in Appendix C.

$TS_{model,protocol,framework}$	TU_{TS}	TA_{TS}	$TCR_{TS} (\sigma)$
Mistral-7B, standard, CrewAI	0.910	0.969	0.667 (0.186)
Mistral-7B, MCP, CrewAI	0.883	0.947	0.633 (0.222)
Mistral-7B, A2A, CrewAI	0.893	0.952	0.628 (0.196)
Mistral-7B, standard, LangChain	0.925	0.983	0.680 (0.207)
Mistral-7B, MCP, LangChain	0.918	0.987	0.659 (0.199)
Mistral-7B, A2A, LangChain	0.910	0.967	0.631 (0.215)
Mistral-7B, standard, LangGraph	0.931	0.980	0.673 (0.187)
Mistral-7B, MCP, LangGraph	0.740	0.778	0.579 (0.181)
Mistral-7B, A2A, LangGraph	0.924	0.979	0.669 (0.223)
Mistral-7B, standard, Semantic Kernel	0.886	0.963	0.634 (0.222)
Mistral-7B, MCP, Semantic Kernel	0.892	0.950	0.642 (0.204)
Mistral-7B, A2A, Semantic Kernel	0.934	0.977	0.670 (0.211)
Qwen2.5-7B, standard, CrewAI	0.830	0.844	0.689 (0.179)
Qwen2.5-7B, MCP, CrewAI	0.942	0.980	0.583 (0.269)
Qwen2.5-7B, A2A, CrewAI	0.942	0.983	0.682 (0.234)
Qwen2.5-7B, standard, LangChain	0.975	0.995	0.795 (0.157)
Qwen2.5-7B, MCP, LangChain	0.952	0.975	0.537 (0.278)
Qwen2.5-7B, A2A, LangChain	0.942	0.984	0.676 (0.263)
Qwen2.5-7B, standard, LangGraph	0.973	0.989	0.780 (0.168)
Qwen2.5-7B, MCP, LangGraph	0.939	0.979	0.509 (0.268)
Qwen2.5-7B, A2A, LangGraph	0.932	0.978	0.676 (0.260)
Qwen2.5-7B, standard, Semantic Kernel	0.977	0.988	0.767 (0.170)
Qwen2.5-7B, MCP, Semantic Kernel	0.942	0.970	0.509 (0.279)
Qwen2.5-7B, A2A, Semantic Kernel	0.947	0.982	0.693 (0.267)
Llama-4-Scout-17B-16E, standard, CrewAI	0.988	0.991	0.412 (0.270)
Llama-4-Scout-17B-16E, MCP, CrewAI	0.664	0.652	0.322 (0.279)
Llama-4-Scout-17B-16E, A2A, CrewAI	0.754	0.721	0.489 (0.271)
Llama-4-Scout-17B-16E, standard, LangChain	0.949	0.948	0.379 (0.278)
Llama-4-Scout-17B-16E, MCP, LangChain	0.671	0.611	0.333 (0.264)
Llama-4-Scout-17B-16E, A2A, LangChain	0.747	0.686	0.464 (0.269)
Llama-4-Scout-17B-16E, standard, LangGraph	0.990	0.992	0.376 (0.283)
Llama-4-Scout-17B-16E, MCP, LangGraph	0.651	0.615	0.314 (0.282)
Llama-4-Scout-17B-16E, A2A, LangGraph	0.753	0.707	0.482 (0.274)
Llama-4-Scout-17B-16E, standard, Semantic Kernel	0.856	0.839	0.759 (0.243)
Llama-4-Scout-17B-16E, MCP, Semantic Kernel	0.814	0.725	0.605 (0.278)
Llama-4-Scout-17B-16E, A2A, Semantic Kernel	0.845	0.767	0.655 (0.266)

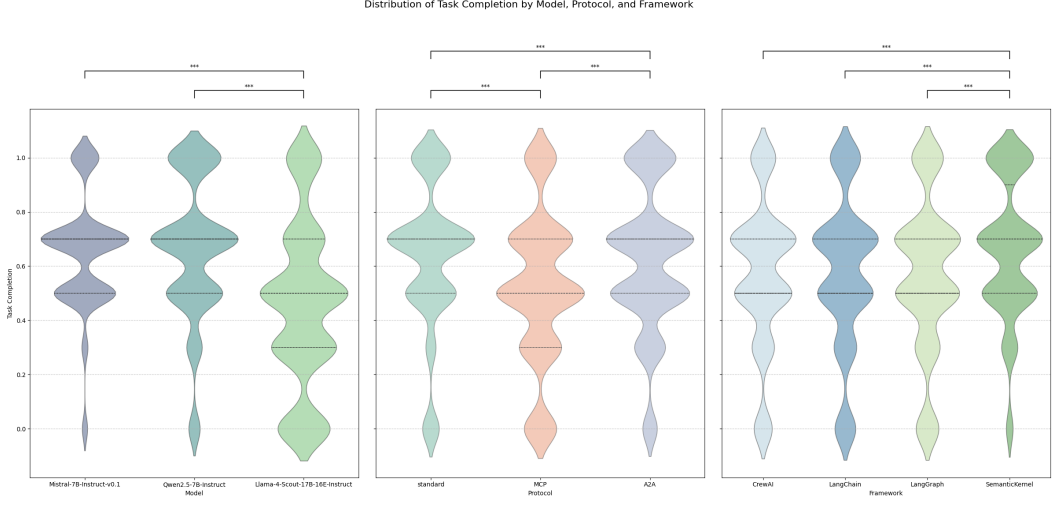


Fig. 2. Task completion scores by model, protocol, and framework. Dunn’s Post-hoc pairwise significant differences are shown according to the following coding: *** for $p - value < 0.001$, ** for $p - value < 0.01$, and * for $p - value < 0.05$.

5.2 Efficiency

Efficiency was proposed as a multi-dimensional profile (Equation 4) that combines task completion score (Equation 1) with convergence time (Equation 2) and verbosity (Equation 3).

5.2.1 Convergence Time. Kruskal-Wallis test for **convergence time** suggests that there are significant differences for model (H-statistic = 81.538, $p - value < 0.001$), protocol (H-statistic = 113.922, $p - value < 0.001$), and framework (H-statistic = 807.034, $p - value < 0.001$).

Dunn’s Post-hoc suggests significant pairwise differences for model, when assessing convergence time, as follows:

- Llama-4-Scout-17B-16E vs. Mistral-7B ($p - value < 0.001$),
- Llama-4-Scout-17B-16E vs. Qwen2.5-7B ($p - value < 0.001$) and
- Mistral-7B vs. Qwen2.5-7B ($p - value < 0.001$).

Dunn’s Post-hoc suggests significant pairwise differences for protocol, when assessing convergence time, as follows:

- A2A vs. MCP ($p - value < 0.001$) and
- MCP vs. standard ($p - value < 0.001$).

Dunn’s Post-hoc suggests significant pairwise differences for framework, when assessing convergence time, as follows:

- CrewAI vs. LangChain ($p - value < 0.001$),
- CrewAI vs. LangGraph ($p - value < 0.001$),
- CrewAI vs. SemanticKernel ($p - value < 0.001$),
- LangChain vs. SemanticKernel ($p - value < 0.001$), and
- LangGraph vs. SemanticKernel ($p - value < 0.001$).

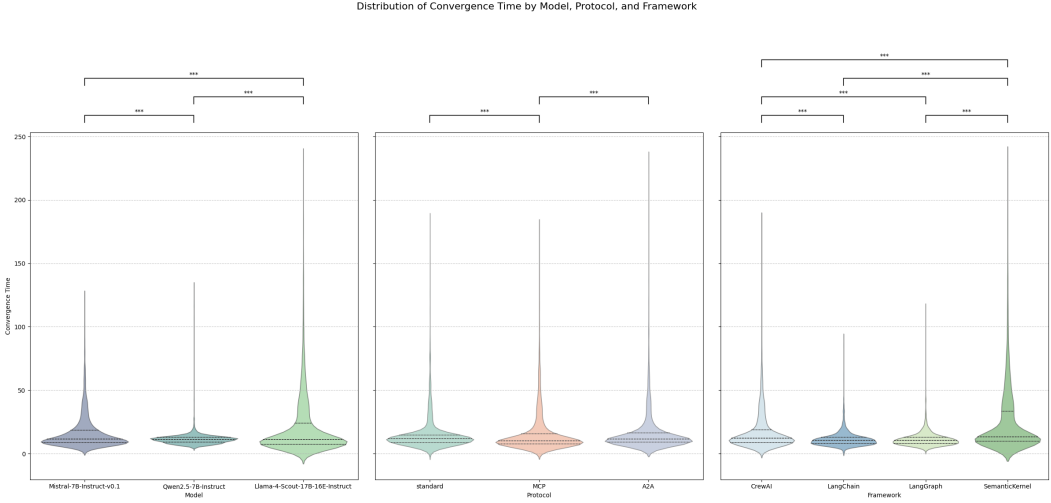


Fig. 3. Convergence time in seconds grouped by model, protocol, and framework. Dunn's Post-hoc pairwise significant differences are shown according to the following coding: *** for $p - \text{value} < 0.001$, ** for $p - \text{value} < 0.01$, and * for $p - \text{value} < 0.05$.

Figure 3 shows a violin plot grouping convergence time by model, protocol, and framework. Considering significant differences found and Kruskal-Wallis mean ranks per group, results suggest that the best-performing technology stack aiming at lowest convergence time should combine: **Qwen2.5-7B** (mean ranks=4974.57), **MCP** (mean rank=4874.13), and **LangChain** (mean rank=4510.52).

5.2.2 Verbosity. Kruskal-Wallis test for **verbosity** suggests that there are significant differences for model (H-statistic = 2483.926, $p - \text{value} < 0.001$), protocol (H-statistic = 1156.463, $p - \text{value} < 0.001$), and framework (H-statistic = 60.922, $p - \text{value} < 0.001$).

Dunn's Post-hoc suggests significant pairwise differences for model, when assessing verbosity, as follows:

- Llama-4-Scout-17B-16E vs. Mistral-7B ($p - \text{value} < 0.001$),
- Llama-4-Scout-17B-16E vs. Qwen2.5-7B ($p - \text{value} < 0.001$), and
- Mistral-7B vs. Qwen2.5-7B ($p - \text{value} < 0.001$).

Dunn's Post-hoc suggests significant pairwise differences for protocol, when assessing verbosity, as follows:

- A2A vs. MCP ($p - \text{value} < 0.001$),
- A2A vs. standard ($p - \text{value} < 0.001$), and
- MCP vs. standard ($p - \text{value} < 0.001$).

Dunn's Post-hoc suggests significant pairwise differences for framework, when assessing verbosity, as follows:

- CrewAI vs. LangChain ($p - \text{value} < 0.001$),
- CrewAI vs. LangGraph ($p - \text{value} < 0.001$), and
- CrewAI vs. SemanticKernel ($p - \text{value} < 0.001$).

Figure 4 shows a violin plot grouping verbosity (in tokens) by model, protocol, and framework. Considering significant differences found and Kruskal-Wallis mean ranks per group, results suggest that the best-performing technology stack aiming at lowest verbosity should combine: **Qwen2.5-7B** (mean ranks=3713.37), **standard** (mean rank=4011.82), and **LangChain** (mean rank=5113.11).

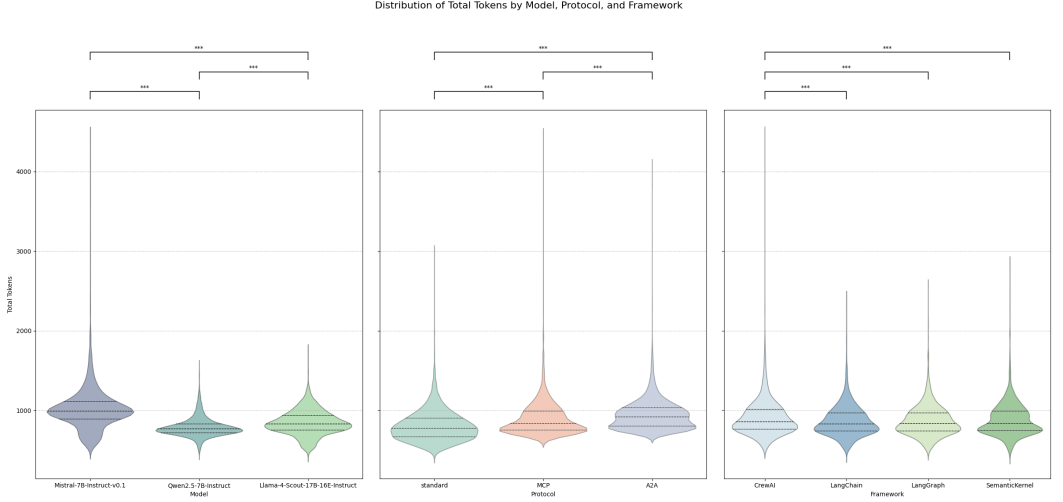


Fig. 4. Verbosity in tokens grouped by model, protocol, and framework. Dunn's Post-hoc pairwise significant differences are shown according to the following coding: *** for $p - value < 0.001$, ** for $p - value < 0.01$, and * for $p - value < 0.05$.

5.2.3 Efficiency. Bearing in mind that efficiency is being assessed as a multi-dimensional efficiency profile composed of task completion, convergence time, and verbosity of agents, results suggest that the technology stack composed of **Qwen2.5-7B**, **standard**, **LangChain** is a promising combination when aiming at high task completion rates while valuing low convergence time and low verbosity (Table 6).

Figure 5 shows a Pareto frontier analysis combining task completion, convergence time, and verbosity. It is possible to identify different stack technologies that lead to slightly different results. Highest task completion scores obtained with **Qwen2.5-7B**, **standard**, **LangChain** and **Qwen2.5-7B**, **standard**, **Semantic Kernel**. Technologies stacks that balance low convergence time with low verbosity are **Qwen2.5-7B**, **A2A**, **Semantic Kernel** and **Qwen2.5-7B**, **A2A**, **LangChain**. Finally, Pareto frontier also shows that technology stacks with promising low convergence time and decent task completion are **Mistral-7B**, **standard**, **LangChain** and **Mistral-7B**, **standard**, **LangGraph**.

Table 3. Mean convergence time (CT_{TS}) per task in seconds and mean verbosity V_{TS} per task in tokens for different technology stacks.

$TS_{model,protocol,framework}$	$CT_{TS}(\sigma)$	$V_{TS}(\sigma)$
Mistral-7B, standard, CrewAI	20.8 (13.3)	912.4 (273.6)
Mistral-7B, MCP, CrewAI	32.8 (19.1)	1186.6 (405.5)
Mistral-7B, A2A, CrewAI	32.8 (17.2)	1176.6 (332.1)
Mistral-7B, standard, LangChain	9.5 (3.2)	822.7 (212.5)
Mistral-7B, MCP, LangChain	13.1 (6.9)	1088.0 (210.1)
Mistral-7B, A2A, LangChain	13.1 (7.8)	1059.9 (177.8)
Mistral-7B, standard, LangGraph	9.3 (2.8)	823.2 (210.5)
Mistral-7B, MCP, LangGraph	13.5 (7.6)	1061.5 (200.8)
Mistral-7B, A2A, LangGraph	12.9 (7.3)	1079.6 (194.7)
Mistral-7B, standard, Semantic Kernel	9.9 (3.7)	828.0 (212.0)
Mistral-7B, MCP, Semantic Kernel	13.1 (8.4)	1082.7 (241.7)
Mistral-7B, A2A, Semantic Kernel	12.7 (6.3)	1065.2 (174.0)
Qwen2.5-7B, standard, CrewAI	11.7 (3.2)	830.6 (168.6)
Qwen2.5-7B, MCP, CrewAI	10.3 (3.7)	764.7 (60.1)
Qwen2.5-7B, A2A, CrewAI	11.2 (6.7)	809.9 (73.7)
Qwen2.5-7B, standard, LangChain	13.1 (2.8)	802.4 (167.5)
Qwen2.5-7B, MCP, LangChain	9.6 (3.0)	749.3 (71.4)
Qwen2.5-7B, A2A, LangChain	11.7 (2.8)	795.2 (83.6)
Qwen2.5-7B, standard, LangGraph	13.4 (2.8)	804.5 (170.2)
Qwen2.5-7B, MCP, LangGraph	9.7 (6.7)	741.8 (57.3)
Qwen2.5-7B, A2A, LangGraph	11.3 (2.9)	802.4 (87.2)
Qwen2.5-7B, standard, Semantic Kernel	13.0 (2.0)	819.6 (173.9)
Qwen2.5-7B, MCP, Semantic Kernel	10.8 (11.5)	749.6 (69.0)
Qwen2.5-7B, A2A, Semantic Kernel	11.2 (2.8)	802.1 (83.5)
Llama-4-Scout-17B-16E, standard, CrewAI	15.7 (15.6)	756.9 (137.4)
Llama-4-Scout-17B-16E, MCP, CrewAI	8.6 (3.8)	856.5 (95.3)
Llama-4-Scout-17B-16E, A2A, CrewAI	11.2 (7.1)	961.7 (111.7)
Llama-4-Scout-17B-16E, standard, LangChain	10.6 (7.7)	726.5 (137.0)
Llama-4-Scout-17B-16E, MCP, LangChain	9.7 (5.6)	834.1 (91.6)
Llama-4-Scout-17B-16E, A2A, LangChain	12.7 (7.6)	944.4 (117.8)
Llama-4-Scout-17B-16E, standard, LangGraph	10.3 (4.9)	735.9 (139.5)
Llama-4-Scout-17B-16E, MCP, LangGraph	9.8 (6.5)	831.2 (82.0)
Llama-4-Scout-17B-16E, A2A, LangGraph	11.5 (4.8)	939.5 (114.1)
Llama-4-Scout-17B-16E, standard, Semantic Kernel	45.8 (25.5)	812.9 (198.6)
Llama-4-Scout-17B-16E, MCP, Semantic Kernel	51.5 (27.8)	860.1 (143.8)
Llama-4-Scout-17B-16E, A2A, Semantic Kernel	58.8 (30.6)	919.1 (189.9)

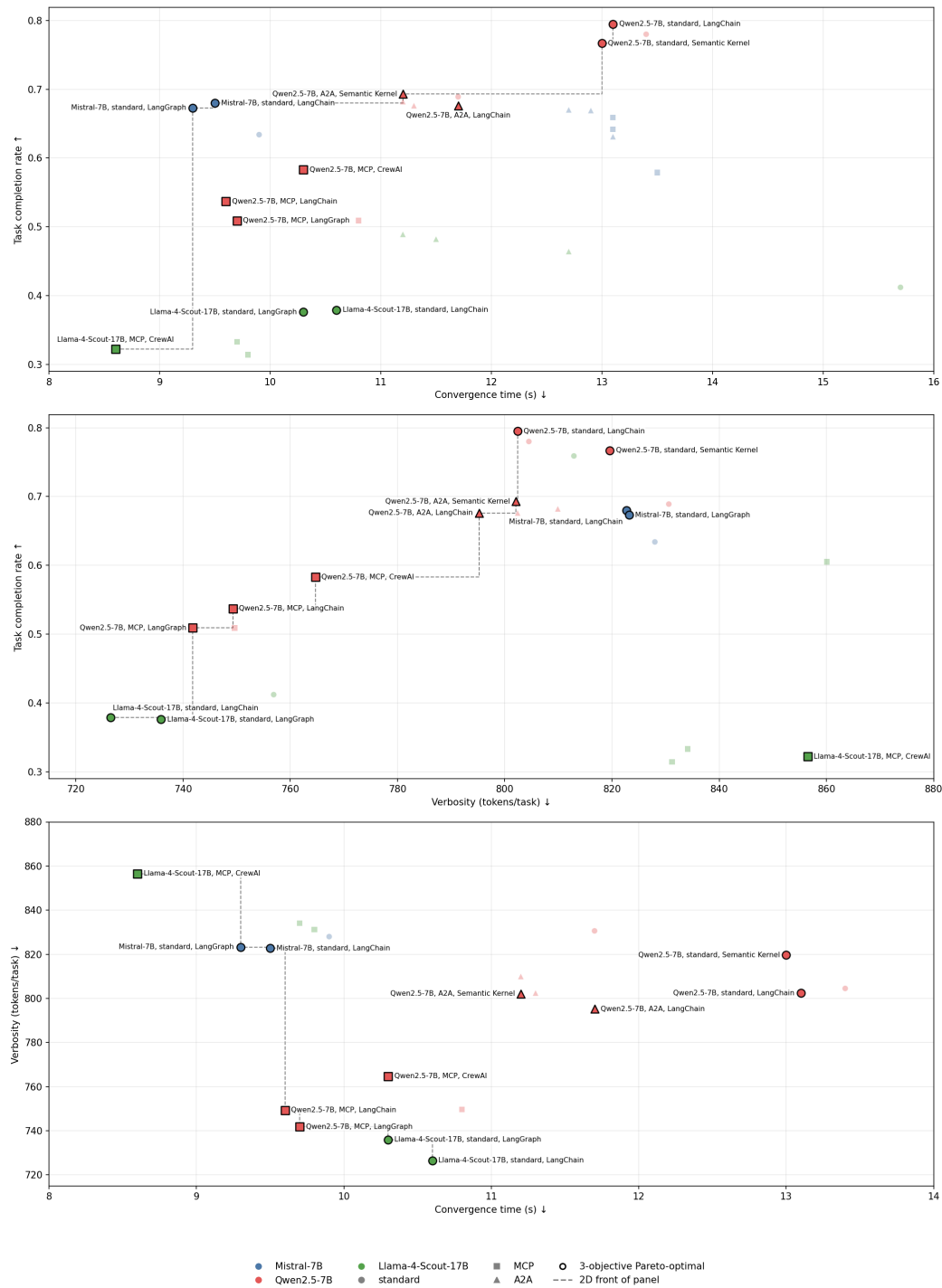


Fig. 5. Pareto analysis zoomed on the frontier region of technology stacks combining task completion, verbosity, and convergence time.

5.3 Tool Use

Tool use was computed according to Equation 5. In the results that follow, the only model that externalized the use of tools in this experiment was Qwen2.5-7B. Hence, the proper analysis combining tool usage by all combinations was affected by model behaviors of answering using internal knowledge or not externalizing the tool use.

Kruskal-Wallis test for tool use rate suggests that there are significant differences for protocol (H-statistic = 11.636, $p - value = 0.003$) and framework (H-statistic = 32.818, $p - value < 0.001$).

Dunn's Post-hoc suggests significant pairwise differences for protocol, when assessing tool usage, as follows:

- A2A vs. MCP ($p - value = 0.002$).

Dunn's Post-hoc suggests significant pairwise differences for framework, when assessing tool usage, as follows:

- CrewAI vs. LangChain ($p - value < 0.001$),
- CrewAI vs. LangGraph ($p - value < 0.001$), and
- CrewAI vs. SemanticKernel ($p - value < 0.001$).

Figure 6 shows a violin plot grouping tool use success by protocol and framework. Considering significant differences found and Kruskal-Wallis mean ranks per group, results suggest that **MCP** (mean rank=1856.70), and **SemanticKernel** (mean rank=1856.14) are a combination that promoted tool use and obtained the best results under the tested conditions.

Figure 6: Distribution of Tool Use Success Rate by Model, Protocol, and Framework

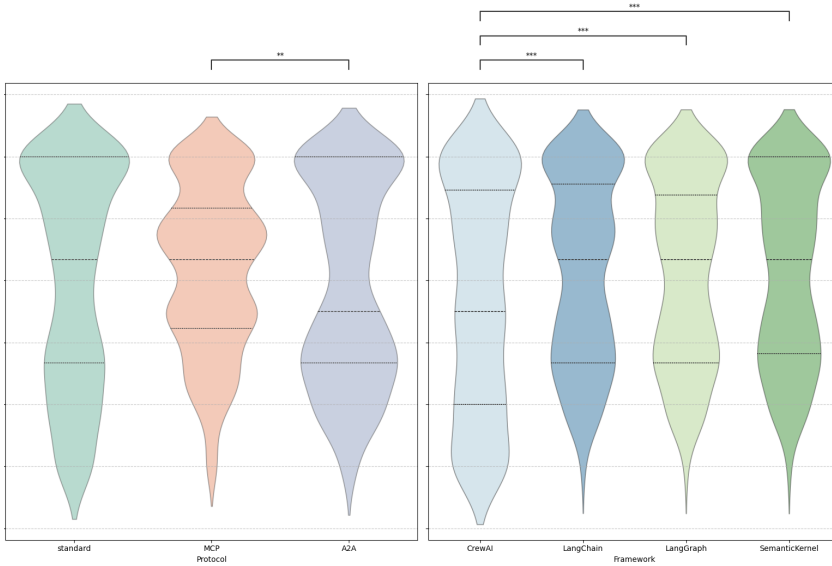


Fig. 6. Tool use success rate for Qwen2.5-7B by protocol and framework. Dunn's Post-hoc pairwise significant differences are shown according to the following coding: *** for $p - value < 0.001$, ** for $p - value < 0.01$, and * for $p - value < 0.05$.

Table 4. Tool use success rate for different technology stacks. Not shown technology stacks did not register tool usage.

$TS_{model,protocol,framework}$	$TUSR_{TS} (\sigma)$
Qwen2.5-7B, standard, CrewAI	0.543 (0.442)
Qwen2.5-7B, MCP, CrewAI	0.523 (0.306)
Qwen2.5-7B, A2A, CrewAI	0.570 (0.330)
Qwen2.5-7B, standard, LangChain	0.638 (0.310)
Qwen2.5-7B, MCP, LangChain	0.670 (0.238)
Qwen2.5-7B, A2A, LangChain	0.577 (0.323)
Qwen2.5-7B, standard, LangGraph	0.632 (0.313)
Qwen2.5-7B, MCP, LangGraph	0.665 (0.229)
Qwen2.5-7B, A2A, LangGraph	0.586 (0.327)
Qwen2.5-7B, standard, Semantic Kernel	0.609 (0.317)
Qwen2.5-7B, MCP, Semantic Kernel	0.681 (0.243)
Qwen2.5-7B, A2A, Semantic Kernel	0.603 (0.317)

5.4 Social Values

Social value stability score was computed according to Equation 8. Kruskal-Wallis test for social value scores suggests that there are significant differences for model (H-statistic = 1081.219, p -value < 0.001), protocol (H-statistic = 6.534, p -value = 0.038), and framework (H-statistic = 73.181, p -value < 0.001).

Dunn's Post-hoc suggests significant pairwise differences for model, when assessing social value stability, as follows:

- Llama-4-Scout-17B-16E vs. Mistral-7B (p -value < 0.001) and
- Llama-4-Scout-17B-16E vs. Qwen2.5-7B (p -value < 0.001).

Dunn's Post-hoc suggests significant pairwise differences for protocol, when assessing social value stability, as follows:

- A2A vs. standard (p -value = 0.032).

Dunn's Post-hoc suggests significant pairwise differences for framework, when assessing social value stability, as follows:

- CrewAI vs. SemanticKernel (p -value < 0.001),
- LangChain vs. SemanticKernel (p -value < 0.001), and
- LangGraph vs. SemanticKernel (p -value = 0.025).

Figure 7 shows a violin plot grouping social value scores by model, protocol, and framework. Considering significant differences found and Kruskal-Wallis mean ranks per group, results suggest that the best-performing technology stack in terms of social value stability should combine: **Mistral-7B** (mean rank=6100.76), **A2A** (mean rank=5460.16), and **SemanticKernel** (mean rank=5778.08).

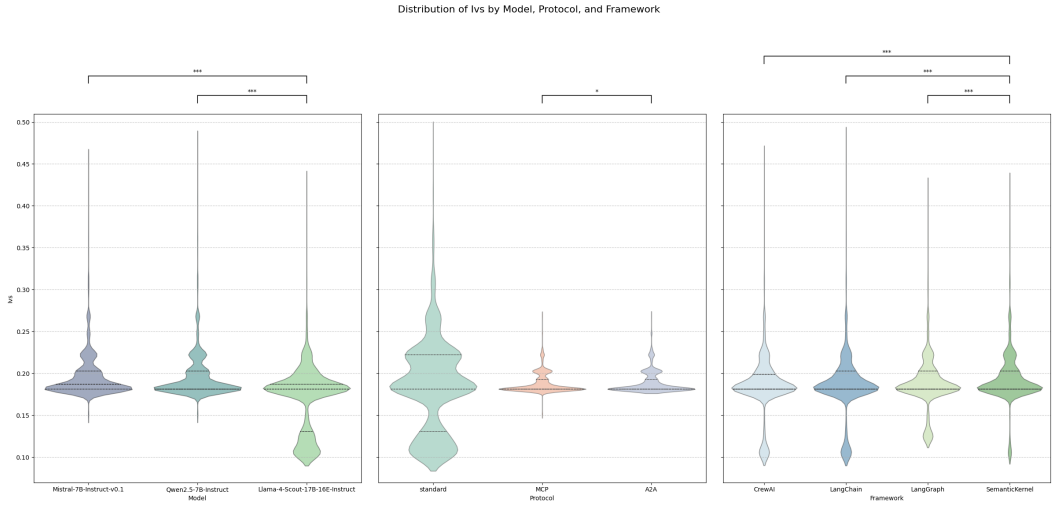


Fig. 7. Social value score by model, protocol, and framework. Dunn’s Post-hoc pairwise significant differences are shown according to the following coding: *** for $p - value < 0.001$, ** for $p - value < 0.01$, and * for $p - value < 0.05$.

Table 5 shows results obtained in terms of social value stability for input task to output, reasoning steps, and social value stability throughout all the steps, from input task up to final answer. It is possible to verify that social values related to input task and output (final answer) are less stable than reasoning steps. Reasoning steps are more stable in the whole process, often representing the same values in its intermediate reasoning steps.

Figure 8 shows the most common social values present in input tasks, reasoning steps, and final answer output. In the figure it is possible to verify that Mind2Web tasks influence the values identified. In reasoning steps, prompts exchanged between agents are stable in terms of value and provide results and content for the next step, highlighted by values such as trust, integrity, safety, and accountability. Social values present in the outputs are also somewhat consistent between models. Transparency, explainability, and right to ownership are present in the top 5 most frequent values for both 7B models shown.

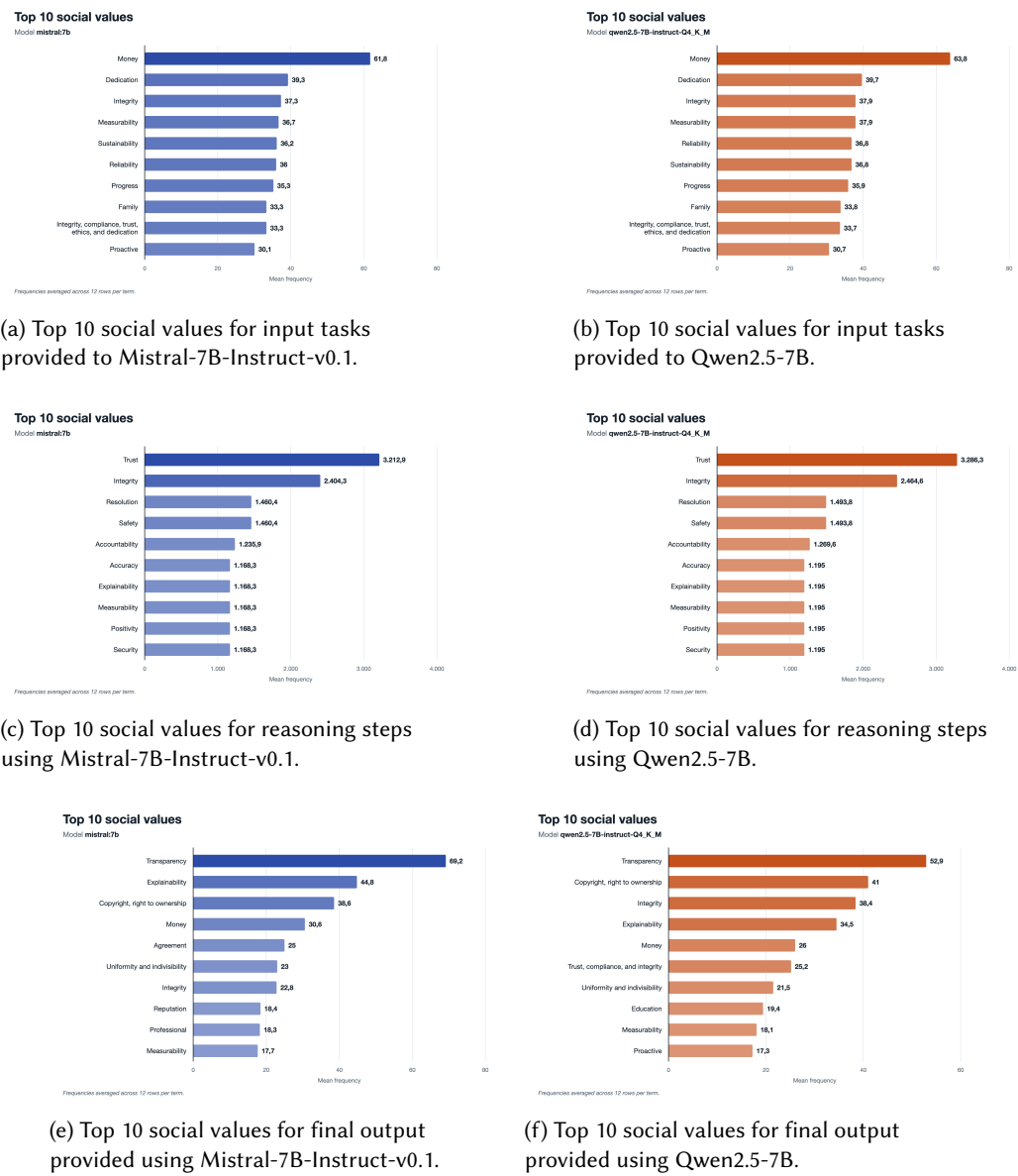


Fig. 8. Comparison of mean frequency of social values retrieved by a human-curated prompt social value classifier [51]. Input tasks (a, b) show that classifier is stable. Reasoning steps (c, d) show that intermediate reasoning steps are consistent reporting agents functioning, regardless of input. Social values represented in the final answer (e, f) are similar, differing in the ranking involving some values.

Table 5. Social value stability considering values present in the prompt and final answer ($SV_{TS_{p,oa}}$), reasoning steps ($SV_{TS_{reasoning}}$), and all steps from prompt, reasoning steps up to final answer (SV_{TS}) for different technology stacks; the higher the better.

$TS_{model,protocol,framework}$	$SV_{TS_{p,oa}}$	$SV_{TS_{reasoning}}$	$SV_{TS} (\sigma)$
Mistral-7B, standard, CrewAI	0.156	0.235	0.212 (0.0380)
Mistral-7B, MCP, CrewAI	0.129	0.228	0.189 (0.0116)
Mistral-7B, A2A, CrewAI	0.105	0.228	0.189 (0.0123)
Mistral-7B, standard, LangChain	0.146	0.235	0.212 (0.0383)
Mistral-7B, MCP, LangChain	0.091	0.228	0.189 (0.0124)
Mistral-7B, A2A, LangChain	0.172	0.228	0.190 (0.0134)
Mistral-7B, standard, LangGraph	0.145	0.237	0.213 (0.0385)
Mistral-7B, MCP, LangGraph	0.123	0.228	0.190 (0.0126)
Mistral-7B, A2A, LangGraph	0.108	0.228	0.190 (0.0126)
Mistral-7B, standard, Semantic Kernel	0.143	0.238	0.213 (0.0395)
Mistral-7B, MCP, Semantic Kernel	0.110	0.228	0.189 (0.0115)
Mistral-7B, A2A, Semantic Kernel	0.113	0.228	0.190 (0.0141)
Qwen2.5-7B, standard, CrewAI	0.134	0.235	0.210 (0.0368)
Qwen2.5-7B, MCP, CrewAI	0.107	0.228	0.188 (0.0114)
Qwen2.5-7B, A2A, CrewAI	0.145	0.228	0.190 (0.0139)
Qwen2.5-7B, standard, LangChain	0.175	0.235	0.211 (0.0379)
Qwen2.5-7B, MCP, LangChain	0.085	0.228	0.188 (0.0121)
Qwen2.5-7B, A2A, LangChain	0.144	0.228	0.190 (0.0135)
Qwen2.5-7B, standard, LangGraph	0.124	0.237	0.211 (0.0370)
Qwen2.5-7B, MCP, LangGraph	0.082	0.228	0.188 (0.0125)
Qwen2.5-7B, A2A, LangGraph	0.117	0.228	0.190 (0.0135)
Qwen2.5-7B, standard, Semantic Kernel	0.141	0.235	0.212 (0.0376)
Qwen2.5-7B, MCP, Semantic Kernel	0.125	0.228	0.188 (0.0126)
Qwen2.5-7B, A2A, Semantic Kernel	0.152	0.228	0.189 (0.0128)
Llama-4-Scout-17B, standard, CrewAI	0.126	0.133	0.110 (0.0101)
Llama-4-Scout-17B, MCP, CrewAI	0.086	0.228	0.188 (0.0128)
Llama-4-Scout-17B, A2A, CrewAI	0.086	0.228	0.189 (0.0132)
Llama-4-Scout-17B, standard, LangChain	0.084	0.133	0.110 (0.0094)
Llama-4-Scout-17B, MCP, LangChain	0.098	0.228	0.188 (0.0126)
Llama-4-Scout-17B, A2A, LangChain	0.087	0.228	0.188 (0.0131)
Llama-4-Scout-17B, standard, LangGraph	0.098	0.158	0.130 (0.0105)
Llama-4-Scout-17B, MCP, LangGraph	0.083	0.228	0.189 (0.0137)
Llama-4-Scout-17B, A2A, LangGraph	0.053	0.228	0.190 (0.0143)
Llama-4-Scout-17B, standard, Semantic Kernel	0.211	0.187	0.187 (0.0551)
Llama-4-Scout-17B, MCP, Semantic Kernel	0.120	0.228	0.188 (0.0123)
Llama-4-Scout-17B, A2A, Semantic Kernel	0.157	0.228	0.189 (0.0121)

6 Discussion

In this section we discuss outcomes obtained according to each of the metrics studied defined in Section 3. Table 6 provides an overview of Agentic Web technology stacks results and associated research questions (RQs).

Table 6. Best-performing technology stacks per metric (consolidated from Sections 5.1 to 5.4) and respective research questions (from RQ1 to RQ4)

	Metric	Model	Protocol	Framework
Efficiency (RQ2) {	Task Completion (RQ1)	Qwen2.5-7B	Standard	SemanticKernel
	Convergence Time	Qwen2.5-7B	MCP	LangChain
	Verbosity	Qwen2.5-7B	Standard	LangChain
	Tool Use Success (RQ3)	Qwen2.5-7B*	MCP	SemanticKernel
	Social Value Stability (RQ4)	Mistral-7B	A2A	SemanticKernel

*Only model that externalized tool calls.

6.1 Task Completion

Bearing mind of RQ1 on task completion rate under the conditions studied, results suggest that **Qwen2.5-7B**, **LangChain** and **standard** conditions represent the best-performing baseline. **Qwen2.5-7B** with other conditions, considering protocol messages MCP and A2A, obtained worse results. However, when using standard messages, it obtained better results. Beyond task completion, Table 2 also shows that models often understood the proposed task and that reasoning steps kept track of the task at hand. Lower values for task understanding and task adherence occurred when combining **Llama-4-Scout-17B-16E** with **MCP** and **A2A**, suggesting that testing how agents understand the task and whether or not internal reasoning steps continue on track has an interesting value in such evaluations. Finally, Table 2 also reveals a pattern in which task completion is impacted under **Qwen2.5-7B** and **MCP** conditions.

6.2 Efficiency

Bearing mind of RQ2 on efficiency, which combines task completion rate, convergence time, and verbosity, results suggest that **Qwen2.5-7B**, **LangChain** and **standard** conditions represent the best-performing balance combining task completion, low convergence time, and low verbosity (Table 6).

Figure 5 presented a Pareto frontier assessment allowing the identification of different nuances in the combination of task completion rates, convergence time, and verbosity. Better task completion scores were obtained under **Qwen2.5-7B**, **standard**, **LangChain** and **Qwen2.5-7B**, **standard**, **Semantic Kernel** conditions. Better balance on low convergence time and low verbosity were obtained under **Qwen2.5-7B**, **A2A**, **Semantic Kernel** **Qwen2.5-7B**, **A2A**, **LangChain** conditions. Finally, technology stacks with low convergence time and average task completion were obtained under **Mistral-7B**, **standard**, **LangChain** **Mistral-7B**, **standard**, **LangGraph** conditions.

Finally, Table 6 also reveals an outlier under conditions **Llama-4-Scout-17B-16E** and **SemanticKernel**, given that they take more time to converge, but, at the same time, are not as verbose as other conditions.

6.3 Tool Use

Bearing mind of RQ3 on tool use, results suggest that **Qwen2.5-7B**, **Semantic Kernel** and **MCP** conditions used more the provided tools. Table 4 and Figure 6 also show that **MCP** was the condition that obtained best-performing tool use success rates, suggesting that MCP messages foment agents to select and use tools more appropriately. However, as presented previously, tool usage was only performed (or externalized) by agents powered by **Qwen2.5-7B** model. Hence, results here are not related to the technology stack, but only under protocol and framework conditions within **Qwen2.5-7B**. More details on this is detailed in the limitations section (Section 7.4).

6.4 Social Value

Bearing in mind RQ4 on social value stability, results suggest that **Mistral-7B**, **A2A**, and **SemanticKernel** conditions were more stable regarding how they process, reason, and answer a given task (Figure 7, Figure 8, and Table 5). Figure 7 shows that input tasks were properly tagged according to Mind2Web tasks, reasoning steps remain stable covering the functioning of the agentic workflow, and that outputs between models have intersecting values.

7 Conclusion

This paper presented an empirical assessment of the Agentic Web technology stack. We have presented that when maintaining the tasks fixed and varying models, frameworks, and protocols, results may differ. In consonance with Ehtesham et al. [22], we have also have shown that no single Agentic Web technology stack suffices for all metrics. Each context must be carefully considered, goal metrics defined, and, finally, tasks must be tested in a systematic way. In this sense, MASTIF has shown to be a valuable tool for standardizing empirical studies for the Agentic Web. In other words, we should consider AI as a normal technology [44] and assess its building blocks for our specific contexts, tasks, and controlling for costs [30].

Regarding research questions defined, conditions using **Qwen2.5-7B** as model, **SemanticKernel** and **LangChain** as implementation frameworks represent interesting starting points for ones planning on testing MAS-based solutions for tasks similar to the ones present in the Mind2Web benchmark.

Tool calling assessment was impacted by how models unpack tasks and select tools to use. This is consonant with literature results that show that different models often differ on how they trigger tool use in MAS [18].

Beyond supporting empirical studies for the Agentic Web, this work also presents a proposal for assessing social value stability within agentic workflows. While Fuse was proposed as a MAS simulation framework for user-mediated social reasoning [58], social value stability assessment provided by MASTIF is, to the best of our knowledge, a novel approach for assessing the entire flow from prompts, passing through reasoning steps, up to the final MAS answer.

In terms of social value stability, conditions using **Semantic Kernel** obtained the best-performing social value stability scores. Results represent a baseline for how MAS decompose task and how agents perform reasoning steps. The proposed score based on Jaccard Pairwise score supported the analysis of tasks, from input to output.

7.1 Lessons Learned

In the development of MASTIF and running this experiment, a common issue was the fact that MAS systems count on multiple points of failure, with low observability, and highly non-deterministic behaviors. These characteristics make developing, running, debugging, and testing everything a Software Engineering ‘*nightmare*.’ Tool use is fragile. After troubleshooting and improving tools descriptions, so that models would identify when to use each tool, not all models used the tools provided. In addition, given that tool call is *non-deterministic*, it is impacted by model behavior, model size, embedded knowledge, temperature parameter, among others. In our initial tests, some models refuse to use tools informed, while others tried to use tools that don’t exist (i.e., *tool hallucination*). In terms of internal knowledge, larger models tend to provide answer using internal knowledge, while smaller models tend to use provided tools, as shown in the literature [18]. Selecting models known to promote tool usage has shown to be of key importance when assessing MAS.

In terms of other factors of tool usage, it is important to provide MAS workflows descriptive information about available tools. In our first attempts, we tried using descriptive information about the tools, such as “This tool does this, this, and that”. However, tools only started to be used in our experiments when we changed its description to guidance-like content, such as, “Use this tool when you need to...” Hence, tool’s descriptions help models on identifying when using (or not) each tool. Hence, this required information impacts the input token burden, which may have an impact on token cost as well.

Regarding running tests using MASTIF, it already deals with multiple types of interruption, including quota errors, inference provider issues, etc. However, experiments using multiple models, protocols, and frameworks can result in huge log files detailing all reasoning and judge assessments. Hence, in experiment designs like the one detailed in this work, we suggest running each model in a separate step to allow easier assessment of log files and results.

7.2 Reproducibility

In terms of reproducibility of the experiment reported in this paper, the Mind2Web dataset is available on HuggingFace¹⁴, the Multi-Agent System Testing Framework (MASTIF) developed is open source and available on GitHub¹⁵, together with all parameters, models, prompts, and agentic frameworks considered detailed in this experiment’s YAML file¹⁶. Hence, we are providing all required information for others to reproduce and/or expand the experiment detailed in this paper.

7.3 Ethics Statement

This paper did not involve human participants and the benchmark of tasks considered is an open dataset, i.e., Mind2Web. Tasks from Mind2Web were sampled in a stratified way aiming at covering different tasks, different task complexities, different domains, and Web contexts.

7.4 Limitations

The most notable limitations involve number of models used, judge, implementation frameworks, and tasks considered. All of these impact the cost to run such experiments [30]. Given that all combinations must be tested, adding tasks, combining more models, and using setups with more than one judge –or more powerful ones– result in a high number of task executions (Equation 9), increasing the cost for running such experiment from hundreds to thousands of dollars. Moreover,

¹⁴<https://huggingface.co/datasets/osunlp/Mind2Web>

¹⁵<https://github.com/cewebbr/mastif/>

¹⁶<https://github.com/cewebbr/mastif/blob/main/experiments/tweb2026.yaml>

the test involving different protocols considers the layer of how different protocols exchange messages. Testing with full implementations would require pre-built servers that provide all protocols studied, something that, to the best of our knowledge, is not available yet on the Web for services that would be representative for such study.

7.5 Next Steps

Results suggest that LLM-based agents and the Agentic Web technology stack should be assessed as any other technology [44], in an empirical and context-sensitive way, covering for multiple possibilities. Hence, next steps of this research include studying the impact of different setups of MAS, including number of agents, hierarchical MAS, different sets of tools, max_steps, temperature, agentic frameworks, new technologies such as WebMCP, and servers implementing MCP and A2A to support full protocol exchanges.

Acknowledgments

This work was supported by the Brazilian Network Information Center (NIC.br). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NIC.br.

References

- [1] 2026. WebMCP. Draft Community Group Report. <https://webmachinelearning.github.io/webmcp/> Published by the Web Machine Learning Community Group.
- [2] A2A. 2025. A2A and MCP: Detailed Comparison. <https://a2a-protocol.org/dev/topics/a2a-and-mcp/>
- [3] AGiXT. 2025. AGiXT. <https://github.com/Josh-XT/AGiXT>
- [4] Agno. 2025. Agno. <https://github.com/agno-agi/agno>
- [5] Agora Project. 2024. Agora Protocol: Scalable Communication for LLM Agents. Open protocol documentation. <https://github.com/agora-protocol>
- [6] Meta AI. 2025. Llama 4: A new era of multimodal intelligence. <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>
- [7] Ken Aizawa. 2025. Writing Effective Tools for Agents—with Agents. Anthropic Engineering. <https://www.anthropic.com/engineering/writing-tools-for-agents> With contributions from colleagues across Anthropic.
- [8] Anthropic. 2024. Model Context Protocol (MCP). Technical specification. <https://github.com/modelcontextprotocol>
- [9] Anthropic. 2025. How we built our multi-agent research system. Technical specification. <https://www.anthropic.com/engineering/multi-agent-research-system>
- [10] Negar Arabzadeh, Siging Huo, Nikhil Mehta, Qingyun Wu, Chi Wang, Ahmed Awadallah, Charles LA Clarke, and Julia Kiseleva. 2024. Assessing and Verifying Task Utility in LLM-Powered Applications. *arXiv preprint arXiv:2405.02178* (2024).
- [11] Debarag Banerjee, Pooja Singh, Arjun Avadhanam, and Saksham Srivastava. 2023. Benchmarking LLM powered Chatbots: Methods and Metrics. *arXiv:2308.04624 [cs.CL]* <https://arxiv.org/abs/2308.04624>
- [12] Kate Blair and Todd Segal. 2025. *ACP Joins Forces with A2A Under the Linux Foundation LF AI & Data*. LF AI & Data Foundation. <https://lfaidata.foundation/communityblog/2025/08/29/acp-joins-forces-with-a2a-under-the-linux-foundations-lf-ai-data/> Describes the merger of the Agent Communication Protocol (ACP) into the Agent2Agent (A2A) protocol to advance open standards for AI agent interoperability; accessed: 2026-03-10.
- [13] Kranti Chalamalasetti, Jana Götz, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. 2023. ClemBench: Using Game Play to Evaluate Chat-Optimized Language Models as Conversational Agents. *arXiv preprint* (2023). *arXiv:2305.13455 [cs.CL]* *arXiv:2305.13455*.
- [14] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. *arXiv preprint arXiv:2504.19413* (2025).
- [15] Minje Choi, Jiaxin Pei, Sagar Kumar, Chang Shu, and David Jurgens. 2023. Do LLMs Understand Social Knowledge? Evaluating the Sociability of Large Language Models with SockET Benchmark. *arXiv:2305.14938 [cs.CL]* <https://arxiv.org/abs/2305.14938>
- [16] Chrome for Developers. 2026. WebMCP. Google Chrome Developers. <https://developer.chrome.com/docs/ai/webmcp> Published May 18, 2026. Last updated October 7, 2026.
- [17] CrewAI. [n. d.]. CrewAI. <https://github.com/crewAIInc/crewAI> Accessed: 2025-09-19.

- [18] Vagner Figueredo de Santana, Diogo Cortiz, Beatriz Rocha, and Henrique S. Xavier. 2026. A Multi-Agent Query Fan-Out Analysis of Popular Google Searches in Brazil. In *Proceedings of the 32nd Brazilian Symposium on Multimedia and the Web* (Lavras/MG). SBC, Porto Alegre, RS, Brasil. <https://sol.sbc.org.br/index.php/webmedia/issue/archive>
- [19] DeepSeek-AI. 2025. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs.CL] <https://arxiv.org/abs/2412.19437>
- [20] Gelei Deng, Yi Liu, Victor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. 2023. PentestGPT: An LLM-empowered Automatic Penetration Testing Tool. arXiv:2308.06782 [cs.SE] <https://arxiv.org/abs/2308.06782>
- [21] Xiang Deng, Yu Gu, Boyuan Zheng, and et al. 2023. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems (NeurIPS)*. <https://arxiv.org/abs/2306.06070>
- [22] Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. 2025. A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP). arXiv:2505.02279 [cs.AI] <https://arxiv.org/abs/2505.02279>
- [23] Adam Fourney, Tyler Payne, Maya Murad, and Saleema Amershi. 2025. Tool-space interference in the MCP era: Designing for agent compatibility at scale. <https://www.microsoft.com/en-us/research/blog/tool-space-interference-in-the-mcp-era-designing-for-agent-compatibility-at-scale/> Accessed: 2025-09-26.
- [24] Stan Franklin and Art Graesser. 1996. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *International workshop on agent theories, architectures, and languages*. Springer, 21–35.
- [25] Benyamin Ghogh, Hadi Nekoei, Aydin Ghogh, Fakhri Karay, and Mark Crowley. 2020. Sampling Algorithms, from Survey Sampling to Monte Carlo Methods: Tutorial and Literature Review. arXiv:2011.00901 [stat.ME] <https://arxiv.org/abs/2011.00901>
- [26] Google DeepMind and collaborators. 2024. Agent-to-Agent (A2A) Protocol. Technical specification and GitHub repository. <https://github.com/agentprotocol/agent-protocol>
- [27] IBM Research. 2024. *BeeAI Framework*. <https://github.com/i-am-bee/beeai-framework>
- [28] Chao Huang Jiabin Tang, Tianyu Fan. 2025. AutoAgent: A Fully-Automated and Zero-Code Framework for LLM Agents. arXiv:202502.05957 [cs.AI] <https://arxiv.org/abs/2502.05957>
- [29] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] <https://arxiv.org/abs/2310.06825>
- [30] Sayash Kapoor, Benedikt Strobl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. 2024. AI Agents That Matter. arXiv:2407.01502 [cs.LG] <https://arxiv.org/abs/2407.01502>
- [31] LangChain. 2025. LangChain. <https://github.com/langchain-ai/langchain>
- [32] LangGraph. 2025. LangGraph. <https://github.com/langchain-ai/langgraph>
- [33] Langroid. 2025. Langroid. <https://github.com/langroid/langroid>
- [34] Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. 2025. ST-WebAgentBench: A Benchmark for Evaluating Safety and Trustworthiness in Web Agents. arXiv:2410.06703 [cs.AI] <https://arxiv.org/abs/2410.06703>
- [35] Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. 2025. Preference Leakage: A Contamination Problem in LLM-as-a-judge. (2025). arXiv:2502.01534 [cs.LG] <https://arxiv.org/abs/2502.01534>
- [36] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- [37] Yuanzhi Liang, Linchao Zhu, and Yi Yang. 2023. Tachikuma: Understanding Complex Interactions with Multi-Character and Novel Objects by Large Language Models. arXiv:2307.12573 [cs.CL] <https://arxiv.org/abs/2307.12573>
- [38] Jiaju Lin, Nicholas Tomlin, Jacob Andreas, and Jason Eisner. 2023. AgentSims: An Open-Source Sandbox for Large Language Model Evaluation. arXiv:2308.04026 [cs.AI] <https://arxiv.org/abs/2308.04026>
- [39] Jerry Liu. 2022. *LlamaIndex*. https://github.com/jerryliu/llama_index
- [40] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023. AgentBench: Evaluating LLMs as Agents. arXiv:2308.03688 [cs.AI] <https://arxiv.org/abs/2308.03688>
- [41] Zhao Mandi, Shreeya Jain, and Shuran Song. 2023. RoCo: Dialectic Multi-Robot Collaboration with Large Language Models. arXiv:2307.04738 [cs.RO] <https://arxiv.org/abs/2307.04738>
- [42] Nikhil Mehta, Milagro Teruel, Patricio Figueroa Sanz, Xin Deng, Ahmed Hassan Awadallah, and Julia Kiseleva. 2023. Improving Grounded Language Understanding in a Collaborative Environment by Interacting with Agents Through Help Feedback. arXiv:2304.10750 [cs.CL] <https://arxiv.org/abs/2304.10750>

- [43] Microsoft / mcp-interviewer. 2025. mcp-interviewer: Catch MCP server issues before your agents do. <https://github.com/microsoft/mcp-interviewer>. Accessed: 2025-09-26, MIT license.
- [44] Arvind Narayanan and Sayash Kapoor. 2025. AI as Normal Technology. Knight First Amendment Institute. <https://knightcolumbia.org/content/ai-as-normal-technology> 25-09 Knight First Amendment Institute.
- [45] OpenAI. 2024. GPT-4o-mini. <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>
- [46] Malte Pietsch, Timo Möller, Bogdan Kostic, Julian Risch, Massimiliano Pippi, Mayank Jobanputra, Sara Zanzottera, Silvano Cerza, Vladimir Blagojevic, Thomas Stadelmann, Tanay Soni, and Sebastian Lee. 2019. *Haystack: the end-to-end NLP framework for pragmatic builders*. <https://github.com/deepset-ai/haystack>
- [47] Agent Communication Protocol Project. 2025. What’s New — Agent Communication Protocol. <https://agentcommunicationprotocol.dev/introduction/whats-new>. Accessed: 2025-09-26.
- [48] PydanticAI. 2025. Pydantic AI. <https://github.com/pydantic/pydantic-ai>
- [49] Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. ‘smolagents’: a smol library to build great agentic systems. <https://github.com/huggingface/smolagents>.
- [50] Stuart Russell and Peter Norvig. 1995. Artificial Intelligence: A Modern Approach.
- [51] Vagner Figueredo de Santana, Sara E Berger, Heloisa Candello, Tiago Machado, Cassia Sampaio Sanctos, Tianyu Su, and Lemara Williams. 2025. Responsible Prompting Recommendation: Fostering Responsible AI Practices in Prompting-Time. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI ’25)*. Association for Computing Machinery, New York, NY, USA, Article 836, 30 pages. doi:10.1145/3706598.3713365
- [52] Timo Schick, Jane Dwivedi-Yu, Roberta Raileanu, Colin Raffel, and Samuel R. Bowman. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/2302.04761>
- [53] Stefan Schneider, Stefan Werner, Ramin Khalili, Artur Hecker, and Holger Karl. 2022. mobile-env: An Open Platform for Reinforcement Learning in Wireless Mobile Networks. <https://github.com/stefanbschneider/mobile-env> GitHub repository and accompanying paper / technical report.
- [54] SemanticKernel. 2025. Semantic Kernel. <https://github.com/microsoft/semantic-kernel>
- [55] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. arXiv:2010.03768 [cs.AI] <https://arxiv.org/abs/2010.03768>
- [56] Significant Gravitas. 2025. AutoGPT. <https://github.com/Significant-Gravitas/AutoGPT>
- [57] Cole Stryker. [n. d.]. Types of AI agents. <https://www.ibm.com/think/topics/ai-agent-types>
- [58] Amir Taubenfeld, Zorik Gekhman, Avigail Grinstein-Dabush, Itay Laish, Ariel Goldstein, Marian Croak, Avinatan Hassidim, Yossi Matias, and Amir Feder. 2026. Verifiable Social Reasoning for LLM Assistants. arXiv:2609.17496 [cs.AI] <https://arxiv.org/abs/2609.17496>
- [59] Qwen Team. 2024. Qwen2.5: A Party of Foundation Models. <https://qwen.ai/blog?id=qwen2.5>
- [60] Jen tse Huang, Man Ho Lam, Eric John Li, Shujie Ren, Wenxuan Wang, Wenxiang Jiao, Zhaopeng Tu, and Michael R. Lyu. 2023. Emotionally Numb or Empathetic? Evaluating How LLMs Feel Using EmotionBench. arXiv:2308.03656 [cs.CL] <https://arxiv.org/abs/2308.03656>
- [61] Ada Defne Tur, Nicholas Meade, Xing Han Lü, Alejandra Zambrano, Arkil Patel, Esin Durmus, Spandana Gella, Karolina Stańczak, and Siva Reddy. 2025. SafeArena: Evaluating the Safety of Autonomous Web Agents. arXiv:2503.04957 [cs.LG] <https://arxiv.org/abs/2503.04957>
- [62] Upsonic. 2025. Upsonic. <https://github.com/upsonic/upsonic>
- [63] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*. <https://arxiv.org/abs/2201.11903>
- [64] Qingyun Wu, Weijia Shi, Shuyuan Zhang, and et al. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *arXiv preprint arXiv:2308.08155* (2023). <https://arxiv.org/abs/2308.08155>
- [65] Binfeng Xu, Xukun Liu, Hua Shen, Zeyu Han, Yuhua Li, Murong Yue, Zhiyuan Peng, Yuchen Liu, Ziyu Yao, and Dongkuan Xu. 2023. Gentopia: A Collaborative Platform for Tool-Augmented LLMs. In *Proceedings of the EMNLP System Demonstrations 2023*. <https://arxiv.org/abs/2308.04030>
- [66] Qiantong Xu, Fenglu Hong, Bo Li, Changran Hu, Zhengyu Chen, and Jian Zhang. 2023. On the Tool Manipulation Capability of Open-source Large Language Models (ToolBench). arXiv:2305.16504 [cs.CL] <https://arxiv.org/abs/2305.16504>
- [67] Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, Weiwen Liu, Ying Wen, Yong Yu, and Weinan Zhang. 2025. A Survey of AI Agent Protocols. arXiv:2504.16736 [cs.AI] <https://arxiv.org/abs/2504.16736>
- [68] Yingxuan Yang, Mulei Ma, Yuxuan Huang, Huacan Chai, Chenyu Gong, Haoran Geng, Yuanjian Zhou, Ying Wen, Meng Fang, Muhao Chen, Shangding Gu, Ming Jin, Costas Spanos, Yang Yang, Pieter Abbeel, Dawn Song, Weinan

- Zhang, and Jun Wang. 2025. Agentic Web: Weaving the Next Web with AI Agents. arXiv:2507.21206 [cs.AI] <https://arxiv.org/abs/2507.21206>
- [69] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2023. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. arXiv:2207.01206 [cs.CL] <https://arxiv.org/abs/2207.01206>
- [70] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*. <https://arxiv.org/abs/2210.03629>
- [71] Zhixin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. 2025. Agent-SafetyBench: Evaluating the Safety of LLM Agents. arXiv:2412.14470 [cs.CL] <https://arxiv.org/abs/2412.14470>
- [72] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2023. WebArena: A Realistic Web Environment for Building Autonomous Agents. arXiv:2307.13854 [cs.AI] <https://arxiv.org/abs/2307.13854>

A Agentic Frameworks

Table 7 lists LLM-agnostic agent frameworks in Python surveyed in this study.

B Benchmarks

Table 8 lists benchmarks surveyed in this study that are related to LLM-based agents.

C Judge Model Rubric

Listing 1 details the rubric used for judge model to assess information retrieved by the Agentic Web technology stacks assessed.

```

1 Evaluate the agent's response and reasoning on the following Mind2Web task.
2
3 Task:
4 {high_level_task}
5
6 Website: {website}
7 Domain: {domain}
8
9 Agent Response:
10 {agent_response}
11
12 Agent Reasoning:
13 {reasoning_steps_text or '(no reasoning provided)'}
14
15 For each of the following dimensions, provide a score from 0.0 to 1.0 and a brief
16   rationale:
17 - task_understanding: how well the agent understood the task goals, constraints,
18   and required output
19 - task_adherence: how well the agent's reasoning stayed on-task and relevant
20 - task_completion: whether the final response is sufficient to complete the task
21   successfully
22
23 Use these anchors:
24 task_understanding:
25 1.0 = Explicitly captures all goals, constraints, and required outputs
26 0.7 = Captures main goal but misses minor constraints or details
27 0.5 = Understands general intent but misses key requirement(s)

```

```
27 0.3 = Misinterprets important parts of the task
28 0.0 = Completely incorrect or unrelated understanding
29
30 task_adherence:
31 1.0 = Fully focused on task, all steps directly relevant
32 0.7 = Mostly on-task, minor irrelevant or redundant steps
33 0.5 = Mixed relevance, some important off-track reasoning
34 0.3 = Frequently off-task or distracted reasoning
35 0.0 = Completely unrelated reasoning
36
37 task_completion:
38 1.0 = Fully completes task with correct and actionable steps/results
39 0.7 = Likely completes task but with minor gaps or inefficiencies
40 0.5 = Partially completes task; important steps missing
41 0.3 = Unlikely to complete task successfully
42 0.0 = Does not complete task at all or is incorrect
43
44 Respond with ONLY a JSON object with keys `task_understanding`, `task_adherence`,
    and `task_completion`.
45 Each value must be an object with keys `score` and `rationale`.
46 Example:
47 {{
48   "task_understanding": {"score": 0.8, "rationale": "..."},
49   "task_adherence": {"score": 0.7, "rationale": "..."},
50   "task_completion": {"score": 0.9, "rationale": "..."}
51 }}
52 Do not include any extra text, markdown, or explanation outside the JSON object.
```

Listing 1. Rubric used by the judge model to assess responses to Mind2Web tasks.

D Generative AI Usage Statement

No Generative AI (GenAI) was used in the *preparation* of this manuscript. GenAI was used though in a *post-writing* step to perform an initial ‘review’ and recommend a bullet list of improvements before the submission (Listing 2).

```
Create an academic paper review for the attached PDF file.
Consider the criteria in the following CFP: {URL}.
Present your feedback in a bullet list, reduced format.
```

Listing 2. Prompt used to perform an initial ‘review’ before submission.

Table 7. LLM-agnostic python agent frameworks (sorted by GitHub stars).

Name	Short description	License	 Stars	 Forks	 Contributors
AutoGPT [56]	Framework for managing GPT-4 agents.	PolyForm 1.0 and MIT	179k	46k	781
LangChain [31]	Framework for building applications through composability.	MIT	133k	21.9k	3,914
AutoGen [64]	Framework for creating MAS that can act autonomously or work alongside humans.	CC-BY-4.0	56.8k	8.5k	558
LlamaIndex [39]	Data framework for LLM application allowinf for covering core and plugin components.	MIT	48.4k	7.2k	1,855
Mem0 [14]	Memory layer for agents.	Apache 2.0	52.2k	5.8k	297
CrewAI [17]	Framework for orchestrating role-playing MAS.	MIT	48.3k	6.6k	302
Agno [4]	Framework for managing MAS in the cloud.	Apache 2.0	39.2k	5.2k	431
LangGraph [32]	MAS orchestration framework for building, managing, and deploying long-running, stateful agents.	MIT	28.6k	4.9k	289
Semantic Kernel [54]	Orchestration framework for MAS	MIT	27.7k	4.5k	435
Smolagents [49]	Model-agnostic framework for MAS aiming for simplicity and multiple modalities.	Apache 2.0	26.5k	2.4k	208
Haystack [46]	Orchestration framework to build customizable applications.	Apache 2.0	24.8k	2.7k	351
CAMEL [36]	MAS framework aiming at large-scale projects.	Apache 2.0	16.6k	1.9k	212
Pydantic AI [48]	Framework for agent workflows inspired by FastAPI development.	MIT	16.2k	1.9k	425
Upsonic [62]	Agent development framework aiming at safety-first agents.	MIT	7.8k	<1k	38
AutoAgent [28]	Automated agent framework enabling creation and deployment of agents using natural language.	MIT	9k	1.3k	5
Langroid [33]	A lightweight, principled framework for extensible creation of LLM-powered MAS.	MIT	4k	<1k	44
BeeAI [27]	Framework for building MAS.	Apache 2.0	3.2k	<1k	70
AGiXT [3]	Framework for orchestration of agents in virtual and physical environments.	MIT	3.2k	<1k	45

Table 8. Benchmarks used in agent evaluations (alphabetical order).

Benchmark	Task	Environment	Modalities
AgentBench [40]	Multi-environment evaluation of LLMs acting as agents	Mixed (APIs, simulated env.)	Actions
AgentEval [10]	Multi-dimensional utility assessment framework for LLM-powered applications	Offline simulation-based task evaluation	Text, Solution traces
Agent-SafetyBench [71]	Tasks to assess safety of agents	Simulated interaction env.	Text
AgentSims [38]	Sandbox for creating agent evaluation tasks	Simulated GUI sandbox	Text, Actions
ALFWorld [55]	Align text-based abstractions with embodied environments	Simulated	Text, Vision, Actions
ClemBench [13]	Game-based interactive tests for chat-optimized LLMs	Simulated games	Text
E2E [11]	End-to-end benchmark and evaluation protocol for chat-bots	Offline evaluation suite	Text
EmotionBench [60]	Evaluate LLMs’ emotional appraisal and empathy via scenarios	Eval framework	Text
Gentopia [65]	Platform for tool-augmented LLM collaboration	Mixed (demos, toolkit)	Text, Tool calls
IGLU [42]	Interactive collaborative environment for grounded language tasks	Collaborative env.	Text, Actions
ST-WebAgentBench [34]	Tasks paired with safety and trustworthiness policies	Simulated interaction env.	Text
Mind2Web [21]	Generalist web agent evaluation via real-world website tasks	Real web browsing	Text, Web actions
Mobile-Env [53]	Coordination environment for wireless mobile networks	Simulated wireless network env.	Actions
Tachikuma [37]	Complex multi-character interactions benchmark for LLMs	Simulated interaction env.	Text
ToolBench [66]	API evaluation suite	API tasks	Text, API calls
PentestGPT [20]	Penetration-testing benchmark and system	Pentest targets	Text, Tool usage
RocoBench [41]	Multi-robot collaboration benchmark	Multi-robot tabletop simulation	Text, Actions
SafeArena [61]	Assesses misuse of web agents towards safety	Simulated interaction	Text
SocKET [15]	Wide-ranging social-language benchmark	Offline tasks	Text
WebArena [72]	Realistic web environment for building autonomous web agents	Real web browsing	Text, Web actions, Tools
WebShop [69]	Large-scale simulated e-commerce environment for grounded agents	Simulated e-commerce site	Text, Web actions, Vision