AgenticPersonas: An Interactive Tool for Natural Language Configuration of Persona Generation

Ismael Garrido-Muñoz[1,*,†], Fernando Martínez-Santiago[1,†] and Arturo Montejo-Ráez[1,†]

[1]CEATIC, Universidad de Jaén, Campus Las Lagunillas, 23071, Jaén, Spain

Abstract

Generating diverse, controlled personas for AI evaluation is vital but challenging due to issues like bias propagation and lack of attribute control, especially with direct LLM use. We present AgenticPersonas, an interactive tool facilitating a novel tag-first generation approach. The tool features an agent guiding users via natural language to create a structured YAML configuration. This configuration drives probabilistic tag generation, ensuring user-defined distributions before an LLM synthesizes natural language personas, aiming to mitigate bias in attribute selection and correlation. The agent leverages LLM-driven insights, context-aware documentation, and configuration adaptation assistance.

Keywords

Personas, Conversational Agents, Bias Mitigation, Interactive Configuration

1. Introduction

The evaluation of Artificial Intelligence (AI) and Machine Learning (ML) systems regarding fairness, robustness, and safety often relies on testing with diverse user personas [1, 2]. However, creating suitable personas presents significant hurdles. Manual creation is resource-intensive, hard to scale [3], may inadvertently encode the creators' own biases. Direct LLM generation risks amplifying societal stereotypes [4] and lacks fine-grained distributional control [5].

Directly employing Large Language Models (LLMs) to generate personas, while seemingly scalable, poses different risks: outputs can be heavily influenced by the models' internal biases [6], potentially amplifying societal stereotypes [4], and achieving fine-grained control over specific attribute distributions (e.g., ensuring representation of minority groups according to precise percentages) is often difficult and requires complex prompt engineering. Ensuring diversity, controlling attributes accurately, and avoiding bias propagation remain key challenges.

A more robust method involves a two-stage framework:

1. **Structured Configuration:** Explicitly define persona attributes (like demographics, roles, interests) and their probability distributions in a structured format (YAML). This allows for complex specifications, such as defining multiple levels of detail for features (‘Geography‘ can appear with different levels of precision, such as ‘country‘, ‘city‘ or ‘region within the country‘), setting probabilities for selecting multiple values (e.g., having a 90% change of generating one race and 10% of 2 races as mixed-race identities with specific property names like ‘mother_race‘, ‘father_race‘), and controlling ranges for numerical attributes like ‘age‘. Generating realistic and unbiased personas for evaluating AI systems is a nontrivial challenge. Prior research indicates that ad hoc or heuristic persona generation techniques can introduce systematic biases and poor coverage of real demographics [7].

SEPLN 2025: 41st International Conference of the Spanish Society for Natural Language Processing, Zaragoza, Spain, 23-26 September 2025.

*Corresponding author.
†These authors contributed equally.
igmunoz@ujaen.es (I. Garrido-Muñoz); dofer@ujaen.es (F. Martínez-Santiago); amontejo@ujaen.es (A. Montejo-Ráez)
0000-0002-6031-031X (I. Garrido-Muñoz); 0000-0002-5345-0902 (F. Martínez-Santiago); 0000-0002-9878-5006 (A. Montejo-Ráez)

2. **Tag-First Generation:** Use this configuration to probabilistically generate structured attribute ‘tags‘ (key-value pairs) for each persona, strictly adhering to the defined distributions before using an LLM to translate these tags into natural language.

This tag-first approach is crucial because it ensures that the fundamental attributes and their frequencies within the generated population are determined by the user’s explicit configuration, not by an LLM’s potentially biased internal knowledge about attribute correlations (e.g., correlations between gender and occupation, or race and socioeconomic status). While the LLM still handles the creative task of generating fluent text, it does so based on a pre-determined, controlled set of attributes. Without structured control, LLMs may default to stereotypical or most likely attribute combinations [8], also prior work [9] has noted that uncontrolled persona generation can result in a narrow set of profiles reinforcing biases

While this framework offers better control, creating the initial structured configuration file manually still requires technical expertise and careful planning. This is where our contribution lies. This paper introduces and demonstrates **AgenticPersonas**, an interactive tool specifically designed to facilitate the critical first stage of this framework – the creation of the structured configuration file – through accessible natural language interaction. AgenticPersonas features a conversational agent that not only guides users through the technical aspects of YAML configuration but also actively assists them in considering bias implications and tailoring the configuration to their specific testing use case.

We demonstrate how AgenticPersonas simplifies defining complex persona configurations, leveraging LLMs for intelligent assistance (insights, suggestions, explanations, updates) and providing contextual help, thereby enabling the generation of tailored, controlled persona sets for rigorous AI evaluation. The focus of this paper is the demonstration of the agent tool itself, its workflow, architecture, and its utility in the persona generation pipeline.

2. The AgenticPersonas Tool

AgenticPersonas serves as an intelligent assistant, bridging the gap between a user’s high-level requirements for persona generation (e.g., “I need personas to test my chatbot for age bias” or “I need to generate people to test a Curriculum Vitae scoring system for Germany.”) and the detailed, structured YAML (a human-readable data format) configuration needed to drive the tag-first generation backend. You can see the workflow in the Figure 1.

2.1. System Overview

The persona creation process involving AgenticPersonas proceeds as follows:

1. **User Need Elicitation:** The user interacts conversationally with the AgenticPersonas agent via a web interface, specifying their goals and context (use case) for persona generation.
2. **Guided Configuration:** The agent assists the user in defining the features via their values, probability distributions, labels, leveraging LLMs to provide insights, explain and make changes based on user feedback. It iteratively builds and refines a YAML configuration file based on the dialogue, with progress visible in the UI.
3. **Configuration Output:** The agent outputs the finalized YAML configuration file, making it available for download. With this file the user can use the created configuration along the included libray to generate any number of personas or even for other unrelated tasks.
4. **Tag Generation (Backend):** This YAML file is input to the tag-first generation module, which samples attribute tags according to the specified setup.
5. **Natural Language Persona Generation (Backend):** The generated tags are passed to an LLM, which synthesizes the final natural language persona descriptions. (Optionally triggered via the agent).

6. **(Optional) Persona Validation:** First the persona tags are validated to find inconsistencies between the tags and after using these tags to generate a Natural Language Persona, a LLM is leveraged to validate tag adherence.
7. **Persona Utilization:** The resulting personas are used for the intended downstream task (e.g., AI system testing, evaluation).

This paper and demonstration focus primarily on the user's interaction with the agent (Steps 1-3) facilitated by the tool's architecture.

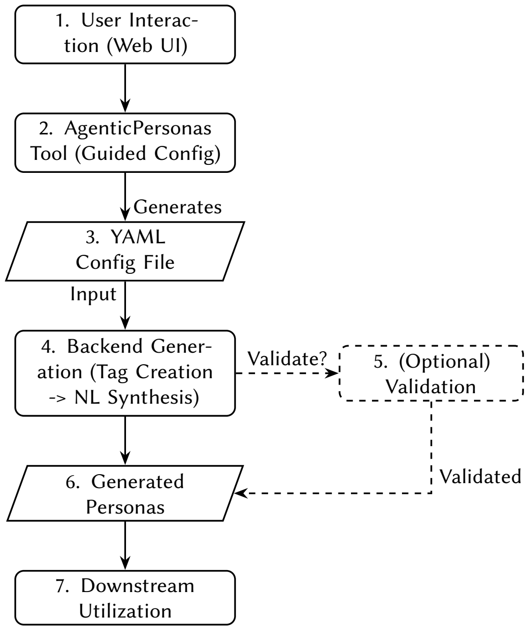

Figure 1: AgenticPersonas agent workflow.

2.2. High-Level Architecture

AgenticPersonas is implemented using a client-server architecture designed for interactive web-based operation (See Figure 2).

- **Client (Browser):** A reactive Single-Page Application (SPA) built with Vue 3 provides the user interface. It features a split-panel layout showing the conversational interaction (chat history, user input field, agent prompts) on one side, and the evolving YAML configuration file along with an optional visualization of the agent's state graph (using Mermaid.js) on the other. The client communicates with the server primarily via an open socket using WebSockets.
- **Server (Backend):** A Python backend built using the FastAPI framework. It serves the Vue.js frontend application and manages WebSocket connections for real-time, bidirectional communication with clients. Upon receiving a request to start, it initiates the core agent logic in a separate thread to avoid blocking asynchronous server operations. The server handles routing messages between the agent thread and the user's browser over the WebSocket.

This architecture separates the presentation layer (client) from the core agent logic and orchestration (server), facilitating development and deployment. The use of WebSockets enables a responsive, conversational experience essential for the agent's interactive nature.

The core agent logic, running on the server, interacts with an external Large Language Model (LLM) service (See Figure 2). While our current implementation primarily utilizes OpenAI's GPT-4o model through its API, the system is designed to be modular. In principle, any sufficiently capable LLM, whether accessed via an API (like models from Anthropic, Google, etc.) or open models (e.g., Llama

3, Mixtral), could be integrated, though the quality and relevance of the agent's assistance (insights, explanations, YAML updates) will naturally vary depending on the chosen model's reasoning and instruction-following abilities.

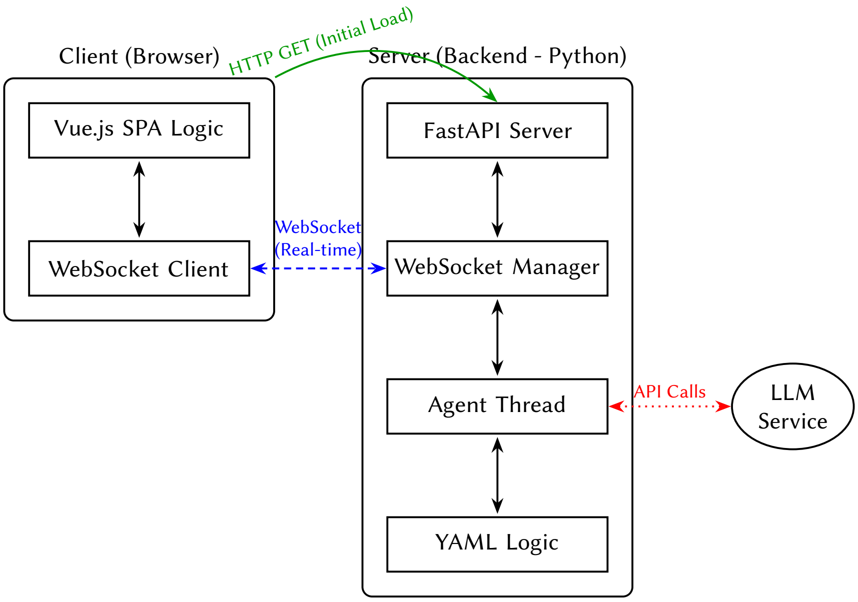

Figure 2: High-level architecture showing the main client-server components and communication paths.

3. Agent Workflow and Implementation

AgenticPersonas facilitates configuration through an interactive, stateful dialogue managed by the LangGraph framework [10]. The agent guides the user from defining their needs to producing a ready-to-use configuration file, leveraging LLMs for several intelligent assistance tasks. The process unfolds as follows:

3.1. Use Case Definition

The interaction begins with the agent prompting the user to describe their specific evaluation goal or "use case" (e.g., "testing a hiring system in Germany," "evaluating a loan application AI for fairness"). This use case provides essential context for all subsequent steps. (Figures 3 and 4)

3.2. Feature Relevance Assessment

The system starts with a default set of configurable persona features (e.g., age, gender, race, geography, ...). Using the provided use case, the agent employs an LLM to classify these features based on their likely relevance:

- *Expected Relevant:* Features likely to directly impact the system being tested (e.g., skills, experience for a hiring tool).
- *Expected Not Relevant:* Features that ideally should not impact the outcome but are critical for evaluation (e.g., race, gender, religion).
- *Not Relevant:* Features unlikely to be pertinent to the specific use case.

This classification helps identify what features (those Expected Not Relevant) need to be in the final configuration for the user's goal.

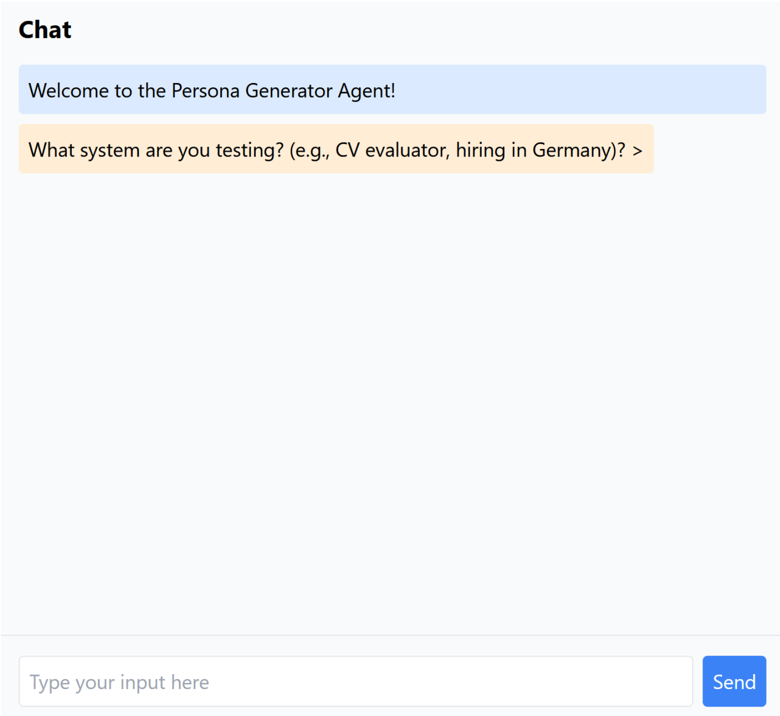

Figure 3: Initial view of the AgenticPersonas interface (left pane), showing the agent's first prompt to the user.

Figure 4: Initial view of the AgenticPersonas interface (right pane), showing the agent's lifecycle. The current step (node) is the one with orange background.

3.3. Contextual Insight Generation

For those selected features identified as Expected Not Relevant, the agent uses an LLM to generate concise, non-obvious "insights." These insights aim to highlight potential bias dynamics or relevant real-world context. For example, given the use case "hiring system in Germany" and the feature "language," an insight might suggest considering languages prevalent in neighboring countries or among significant immigrant communities in Germany, encouraging a more nuanced configuration than just official languages. These insights inform subsequent configuration steps.

3.4. Iterative Feature Configuration

The agent then proceeds feature by feature through the relevant categories identified earlier. For each feature, the cycle involves the following steps.

3.4.1. Initial Adaptation & Explanation

The agent takes the default configuration for the feature and prompts an LLM to propose an initial adaptation tailored to the use case and informed by any generated insights. The goal is to create a starting point that is more relevant and better suited for bias testing than the generic default. This proposed configuration is displayed to the user as YAML. Concurrently, an LLM generates a natural language explanation of the proposed YAML settings (e.g., detailing value probabilities or range definitions). See Figures 5 and 6.

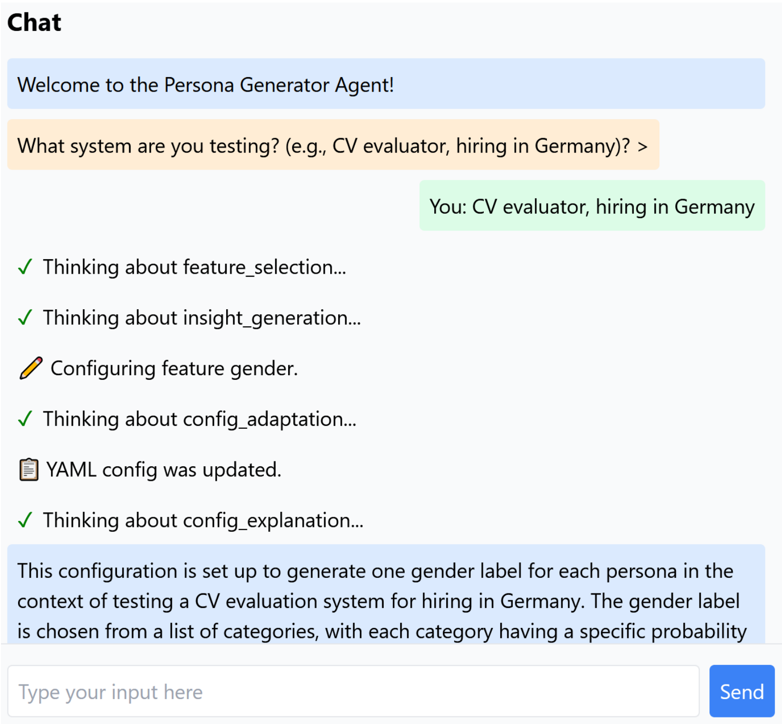

Figure 5: Agent adapting the configuration for a feature based on the use case, showing intermediate processing messages in the UI.

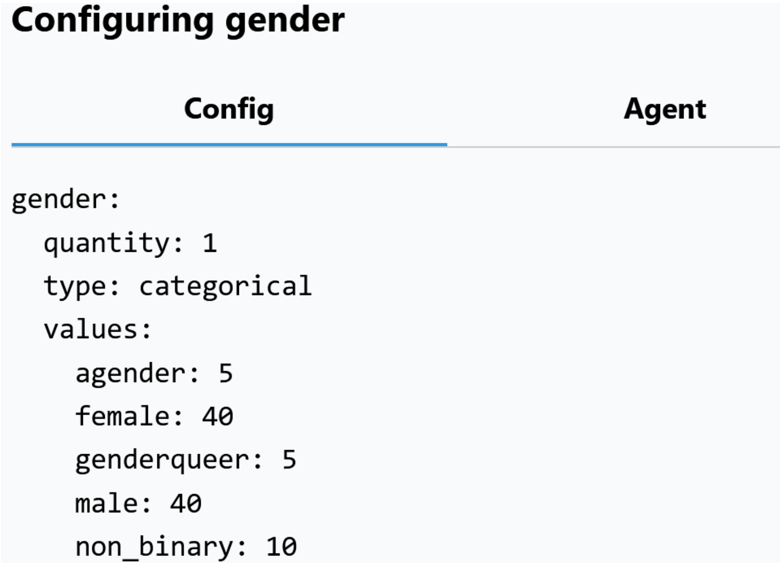

Figure 6: The proposed YAML configuration for a feature displayed in the UI (User interface) before user feedback.

3.4.2. User Feedback and Refinement

The user reviews the proposed configuration and explanation. The user can then:

- *Accept it* and type "continue" to move to the next feature. The UI features a clickable 'continue' button as well.
- *Ask for "help"* to see relevant documentation about the configuration schema.
- *Provide natural language feedback* to request specific changes (e.g., "Add non-binary to gender with 5% probability," "Focus geography on major European capitals," or for the `voted_party` feature, "Add the top 10 on the country with their respective known voting distribution").

3.4.3. Configuration Update

If the user requests changes, the agent uses an LLM to update the YAML configuration according to the request, ensuring adherence to the schema rules (knowledge of which is embedded in the prompt). The updated YAML and a fresh natural language explanation are presented back to the user. See Figure 7.

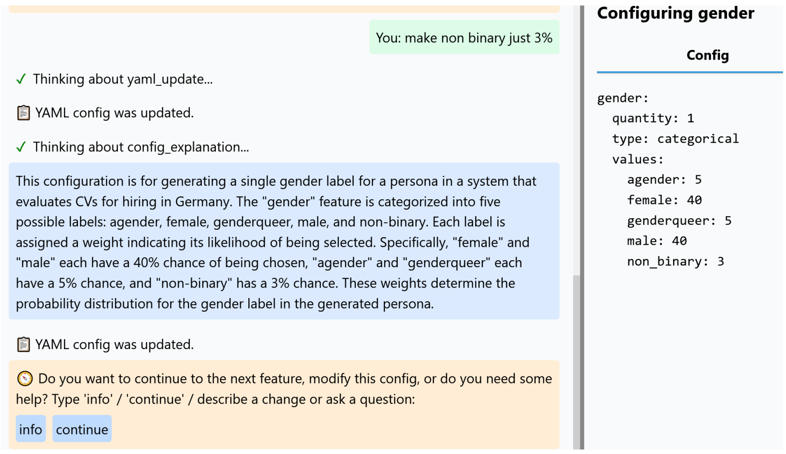

Figure 7: Result of user feedback: The agent provides an explanation and displays the updated YAML configuration in the UI.

This cycle of presenting, explaining, getting feedback, and updating continues until the user indicates satisfaction with the configuration for the current feature. A diagram of this core loop is shown in Figure 8.

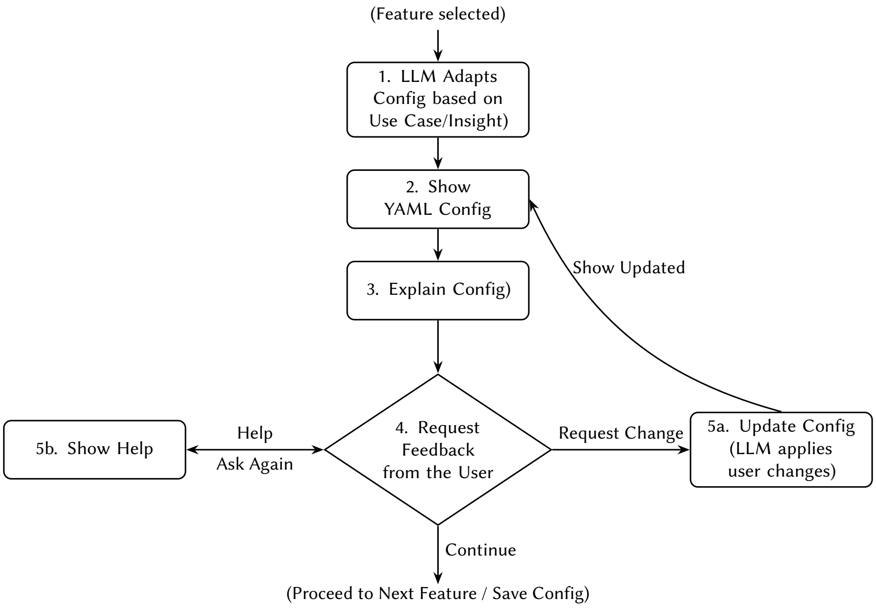

Figure 8: Core loop for configuring a single feature within AgenticPersonas.

3.5. Finalization and Persona Generation

Once all selected features have been configured, the agent saves the complete, tailored YAML file and makes it available for download. At this point, the agent also asks the user if they wish to immediately use this configuration to generate one or multiple personas via the backend (triggering the Tag-First generation process). See Figure 9.

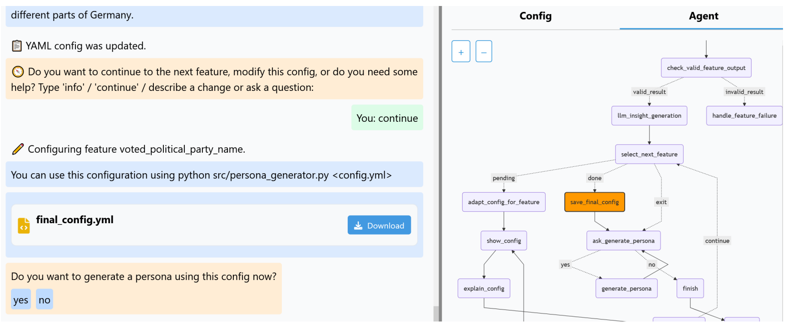

Figure 9: Final stage: A link to download the configured YAML file is provided, and the agent asks whether to proceed with persona generation, offering interactive buttons.

Throughout this workflow, the agent acts as a facilitator, using LLMs to handle complex tasks like understanding context, suggesting relevant settings, adhering to a complex configuration schema, explaining technical details simply, and modifying structured data based on natural language commands, while keeping the user in control of the final output.

4. Use Cases

AgenticPersonas, by simplifying the creation of detailed and statistically controlled configurations, directly enables more rigorous and accessible evaluation of AI systems. The personas generated based on its output are particularly valuable for:

- **Auditing Algorithmic Bias:** Systematically generating personas with specific distributions across sensitive attributes (e.g., gender, race, age, geography, religion, political leaning) allows for targeted testing of AI systems like recruitment tools, loan application processors, content moderation systems, or chatbots. Testers can measure differential treatment or performance disparities based on these controlled attributes.
- **Evaluating Fairness in Recommendation Systems:** Creating user profiles with diverse, explicitly defined characteristics and preferences (facilitated by the agent's configuration guidance) helps audit recommendation engines (e.g., for news, products, jobs) for fairness issues such as exposure inequality or filter bubble effects across different demographic groups.
- **Robustness Testing:** Generating personas representing edge cases or combinations of attributes uncommon in typical datasets helps assess the robustness and reliability of AI systems when faced with less frequent user profiles.
- **Structured Red Teaming:** Employing personas specifically configured to represent vulnerable groups or trigger known sensitivities allows for structured red teaming efforts aimed at proactively discovering hidden biases, stereotypes, or safety failures in AI models before deployment.
- **Synthetic Data Generation for Testing:** When real-world data is scarce, sensitive, or lacks diversity, AgenticPersonas can configure the generation of balanced or specifically skewed synthetic datasets of user profiles or interactions, enabling controlled experiments to isolate and measure algorithmic bias.
- **Developing Standardized Fairness Benchmarks:** The tool can be used to create shareable, reproducible YAML configurations, facilitating the development of standardized persona-based benchmark suites for comparing fairness properties across different AI models or platforms.

In essence, AgenticPersonas empowers researchers and practitioners to move beyond ad-hoc or overly simplistic persona creation, enabling targeted, reproducible, and statistically grounded evaluations essential for developing more fair and reliable AI systems.

5. Limitations

A primary limitation of AgenticPersonas is its heavy dependence on the underlying LLM. The agent's overall effectiveness hinges on the model's ability to accurately interpret natural language, adhere to complex instructions including schema constraints and YAML formatting, and generate relevant insights. Failures in this process can result in incorrect configurations that demand user intervention, particularly since current error handling is limited to simple retries or fallbacks. This reliance also introduces challenges in correctly interpreting complex or ambiguous user requests, which may necessitate multiple clarification turns. Furthermore, while the tag-first generation process mitigates bias in attribute distribution, the conversational LLM assistant could inadvertently introduce subtle biases when shaping the configuration, requiring careful prompt design and oversight. Finally, the system's scope is currently constrained, focusing exclusively on attributes for textual descriptions rather than multi-modal aspects like images, and its prototype status means it lacks production-ready features such as robust user management and advanced security.

6. Future Work

To address these limitations and expand the system's capabilities, our future work will proceed along several strategic avenues. A central focus will be on enhancing the core AI interaction by refining prompt engineering, exploring fine-tuning strategies, and developing more sophisticated error recovery mechanisms to increase robustness. This will be complemented by improving dialogue management to more effectively handle complex constraints and multi-turn refinements. We also aim to increase the system's flexibility and breadth by exploring methods for dynamic schema extension, which would allow users to define new attributes through conversation, and by significantly expanding the library of pre-defined configurations to cover more domains. Further enhancements will include implementing advanced validation logic to check for internal consistency within configurations and investigating multi-modal integration to extend the framework to parameters for generating synthetic images or voice characteristics.

7. Conclusion

AgenticPersonas presents an interactive, agent-based tool designed to tackle the significant challenge of configuring diverse and statistically controlled personas for effective AI system evaluation. By employing a conversational agent built on LangGraph and powered by LLMs, it demystifies the creation of complex YAML configurations required for nuanced persona generation. The agent intelligently guides users through natural language interaction, leveraging context-aware assistance, LLM-generated insights relevant to bias testing, configuration suggestions based on internal documentation, and clear explanations of technical settings.

Crucially, the configurations produced by AgenticPersonas enable a tag-first generation methodology. This ensures that the distribution of core persona attributes is explicitly controlled according to user specifications before an LLM synthesizes the final natural language description, thereby aiming to mitigate the risk of inheriting distributional biases directly from the language model during attribute selection.

By lowering the technical barrier to entry and actively assisting users in navigating complex configuration options and considering bias implications, AgenticPersonas facilitates the creation of tailored, reproducible, and statistically grounded persona sets. This contribution aims to empower researchers and practitioners to conduct more rigorous, accessible, and reliable evaluations, ultimately fostering the development of AI systems that are fairer, more robust, and safer for diverse user populations.

The tool source code can be found in https://github.com/IsGarrido/Gender_Agent_Frozen.

Acknowledgements

This work is funded by the Ministerio para la Transformación Digital y de la Función Pública and Plan de Recuperación, Transformación y Resiliencia - Funded by EU – NextGenerationEU within the framework of the project Desarrollo Modelos ALIA. This work has also been partially supported by Project CONSENSO (PID2021-122263OB-C21), Project MODERATES (TED2021-130145B-I00) and Project SocialTox (PDC2022-133146-C21) funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR.

Declaration on Generative AI

By using the activity taxonomy in ceur-ws.org/genai-tax.html:
During the preparation of this work, the author(s) used Gemini 2.5 Pro (web version) and Copilot (integrated in VS Code). The tools were used in order to: Improve writing style, Grammar and spelling check (Gemini); and to incorporate minor suggestions into the source code (Copilot). After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] I. Garrido-Muñoz, A. Montejo-Ráez, F. Martínez-Santiago, L. A. Ureña-López, A survey on bias in deep NLP, Applied Sciences 11 (2021). URL: https://www.mdpi.com/2076-3417/11/7/3184. doi:10.3390/app11073184.

[2] N. Mehrabi, F. Morstatter, N. A. Saxena, K. Lerman, A. G. Galstyan, A survey on bias and fairness in machine learning, ACM Computing Surveys (CSUR) 54 (2019) 1 – 35. URL: https://api.semanticscholar.org/CorpusID:201666566.

[3] J. Salminen, S. Jung, B. J. Jansen, The future of data-driven personas: A marriage of online analytics numbers and human attributes, in: Proceedings of the 21st International Conference on Enterprise Information Systems - Volume 1: ICEIS, INSTICC, SciTePress, 2019, pp. 608–615. doi:10.5220/0007744706080615.

[4] E. M. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell, On the dangers of stochastic parrots: Can language models be too big?, in: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 610–623. URL: https://doi.org/10.1145/3442188.3445922. doi:10.1145/3442188.3445922.

[5] I. D. Raji, A. Smart, R. N. White, M. Mitchell, T. Gebru, B. Hutchinson, J. Smith-Loud, D. Theron, P. Barnes, Closing the AI accountability gap: defining an end-to-end framework for internal algorithmic auditing, in: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, FAT* '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 33–44. URL: https://doi.org/10.1145/3351095.3372873. doi:10.1145/3351095.3372873.

[6] E. Sheng, K.-W. Chang, P. Natarajan, N. Peng, The woman worked as a babysitter: On biases in language generation, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 3407–3412. URL: https://aclanthology.org/D19-1339/. doi:10.18653/v1/D19-1339.

[7] A. Li, H. Chen, H. Namkoong, T. Peng, LLM Generated Persona is a Promise with a Catch, arXiv e-prints (2025) arXiv:2503.16527. doi:10.48550/arXiv.2503.16527. arXiv:2503.16527.

[8] A. Liu, M. Diab, D. Fried, Evaluating large language model biases in persona-steered generation, in: L.-W. Ku, A. Martins, V. Srikumar (Eds.), Findings of the Association for Computational Linguistics: ACL 2024, Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 9832–9850. URL: https://aclanthology.org/2024.findings-acl.586/. doi:10.18653/v1/2024.findings-acl.586.

[9] L. Sun, T. Qin, A. Hu, J. Zhang, S. Lin, J. Chen, M. Ali, M. Prpa, Persona-L has Entered the Chat: Leveraging LLM and Ability-based Framework for Personas of People with Complex Needs, arXiv e-prints (2024) arXiv:2409.15604. doi:10.48550/arXiv.2409.15604. arXiv:2409.15604.

[10] LangChain, LangGraph, 2024. URL: https://github.com/langchain-ai/langgraph. Accessed: 2025-04-01.