Sim.API: A Middleware to Simplify the Use of User Simulators for Shared Tasks in Conversational Search

Marcel Gohsen
Bauhaus-Universität Weimar
Weimar, Germany

Nailia Mirzakhmedova
Bauhaus-Universität Weimar
Weimar, Germany

Zahra Abbasiantaeb
University of Amsterdam
Amsterdam, The Netherlands

Johannes Kiesel
GESIS - Leibniz Institute for the Social Sciences
Cologne, Germany

Simon Lupart
University of Amsterdam
Amsterdam, The Netherlands

Jeffrey Dalton
University of Edinburgh
Edinburgh, Scotland, UK

Benno Stein
Bauhaus-Universität Weimar
Weimar, Germany

Mohammad Aliannejadi
University of Amsterdam
Amsterdam, The Netherlands

Abstract

Shared tasks offer excellent opportunities for advancing scientific fields and standardizing evaluation methods through collaborative community efforts. With the recent paradigm shift in information retrieval from keyword-based to conversational search, shared tasks can serve as promising sources for standardized evaluation methods in this domain. In particular, user simulation could develop into such an evaluation standard, since it overcomes many of the shortcomings of the Cranfield-paradigm for conversational search. However, employing user simulators in a shared task poses technical and infrastructural challenges, in part because the participants' systems and the organizer-provided simulators must be "online" at the same time, and the interactions between them must be recorded in a uniform format (run submission) in order to carry out the evaluation of the submissions. In this paper, we present Sim.API, a middleware-based run submission system that exposes user simulators through simple API endpoints and records run information without having to rely on run files. This resource aims to simplify the run submission for participants and organizers of shared tasks in conversational search. The source code, resources, and documentation are available in a public repository.[1]

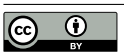
CCS Concepts

• **Computing methodologies** → **Simulation environments**; • **Information systems** → *Evaluation of retrieval results*; *Users and interactive retrieval.*

Keywords

user simulation, conversational search evaluation, shared task infrastructure

[1] https://github.com/marcel-gohsen/user-simulation-api



This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGIR '26, Melbourne, VIC, Australia
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3808599

ACM Reference Format:
Marcel Gohsen, Nailia Mirzakhmedova, Zahra Abbasiantaeb, Johannes Kiesel, Simon Lupart, Jeffrey Dalton, Benno Stein, and Mohammad Aliannejadi. 2026. Sim.API: A Middleware to Simplify the Use of User Simulators for Shared Tasks in Conversational Search. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3805712.3808599

1 Introduction

The conversation-based interaction paradigm seems to have already taken over or will soon take over the traditional keyword-based interaction paradigm as the dominant approach to searching. According to a recent McKinsey report [25], for 44% of users, commercial conversational search tools, such as Perplexity, Google AI, or ChatGPT, are the preferred sources of information to inform purchasing decisions, while only 31% of the users prefer ad hoc search such as Google or Bing. Despite the popularity of conversational search, the question of how to evaluate the effectiveness of these search systems remains open. Established evaluation paradigms for ad hoc retrieval, such as the Cranfield paradigm, transfer poorly into the domain of conversational search, as interactions in multiple turns and the associated dynamically evolving information needs of the user are not taken into account.

In an effort to advance the field of conversational search and to standardize the evaluation of these retrieval systems through contributions of the IR community, various shared tasks have been conducted. One of the earliest was the TREC Conversational Assistance Track (CAsT) taking place annually from 2019 to 2022 [11–13, 21], which was centered around retrieving documents from a collection based on a given information-seeking dialogue. The synthesis of a text-based response based on the retrieved documents was only considered in the last iteration of TREC CAsT, which then evolved into the TREC Interactive Knowledge Assistance Track (iKAT) conducted annually from 2023 to 2025 [2–4]. TREC iKAT focuses on conversational search systems that adapt its responses to the preferences and properties of its users (personalization). SCAI-QReCC [26] is another instance of a shared task dedicated to conversational question answering with a focus on question rewriting, passage retrieval, and answer generation.

The recent iteration of TREC iKAT [4] and the upcoming TREC User Simulation Track 2026[2] signal a focus on an emerging paradigm for the evaluation of conversational search systems: user simulation. The evaluation of systems with the help of user simulations involves (1) imitating the interactions of users with a computer program and (2) using the resulting interaction data as the basis for the evaluation. However, employing user simulators in a shared task presents a challenge, since participants have to interact with the provided simulators in "real-time" in order to generate useful interactions that can be evaluated.

There are two common approaches to address the challenge: run file submissions and software submissions. To use run file submissions in a conversational search shared task with user simulators means that a participant downloads the simulator to their own computer (e.g., as a software package or framework), interact with the simulator through a given interface, record the conversations, and produce a run file that adheres to the organizers' format requirements. Software submissions, on the other hand, require a participant to develop a reusable software artifact that organizers execute on their hardware to produce interaction data for evaluation. Both approaches are prone to errors, for example due to incorrectly formatted run files or crashes of the submitted participant systems on unseen hardware and data. Run files also require the organizers to evaluate on a public test set which can be abused by the participants for training purposes,[3] or the local user simulator can be manipulated to gain an unfair advantage in the competition. Furthermore, software submissions (especially through containerization with Docker) significantly increase the level of technical knowledge required to participate.

In this paper we present Sim.API, a middleware for connecting user simulators with participant systems for shared tasks in conversational search. With Sim.API, organizers of shared tasks can (1) grant participants access to user simulators with secret implementations, (2) query the participating systems based on topics from a secret test dataset, and (3) record conversations with the systems on the organizers' side to ensure fair behavior. Participants complete run submission by using combinations of two basic HTTP requests, which substantially lowers the barrier of participation. Our API is equipped with various quality-of-life features for organizers such as authentication, session and metadata tracking, configurable run and request budgets for participants, and implements the ACID principles [17] to guarantee the validity of run data. An initial practical deployment as part of the TREC iKAT 2025 shared task demonstrated the robustness of the system, which required only minimal support effort on the part of the organizers and resulted in only one support request from the participants.

2 Background and Related Work

A shared task is a collaborative laboratory experiment to evaluate state-of-the art computational solutions to a problem [16]. Shared tasks in conversational search follow a fairly similar procedure: participants of a shared task are asked to develop a conversational search system that retrieves relevant documents from a collection and synthesizes a text-based response to user utterances in multiple turns of interactions. These interactions (i.e., conversations) are then used by organizers to evaluate the effectiveness of the participant systems.

Whereas previously, the user utterances and context were provided by a test dataset of conversations (i.e., participant systems had to predict the next utterance based on a given conversation history), there has recently been a trend toward simulating user utterances, so that conversations are generated through alternating system and simulated user utterances based on a test dataset of topics. This task design for conversational search was first adopted by the Touché Retrieval-Augmented Debating (RAD) task [18], in which the participants developed debate systems that engage with a simulated user in a debate about a controversial claim by retrieving arguments from a predefined collection.

2.1 Submission Formats

The inclusion of user simulators for the evaluation of conversational search systems in shared tasks raises the infrastructural question of how organizer-provided simulators interact with participant systems to produce conversations. The two approaches that have been used in practice are run file submissions and software submissions.

Run File Submissions. A run file is a text file (e.g., formatted as `.json`) that contains conversations between a simulated user and the participant system, document rankings, and other metadata about the run. It is generated by a participating team by installing the user simulation software from the organizers, which is then used to interact with the retrieval system of the participants. Run files are transferred to the organizers as part of the submission process. The interface between user simulators and the participants' systems can be abstracted by a framework to reduce the complexity of participation. Frameworks that define interfaces between user simulators and conversational search systems include SimIIR [5, 20, 28], UserSimCRS [1] and GenIRSim [19].

Run file submission is the standard format of TREC shared tasks and has been used for several years for conversational search tasks such as TREC iKAT '23 and '24 [2, 3] (but without the use of user simulators). A shared task in the domain of conversational search that used this submission format was the micro shared task of the 2025 Simulation for Information Access Workshop (Sim4IA'25) [24]. However, no interactions between simulators and retrieval systems were submitted or recorded, but only simulated utterances based on a given conversation history.

Table 1 compares the advantages of a run file submission format with software submissions and API-based submissions as enabled by Sim.API. Run file submissions probably require the least amount of technical skills from the participants as only a file needs to be submitted, and participants typically have no restrictions on the software they are allowed to use as long as there are no conflicts with the framework. However, run file submissions are especially prone to errors as the run files can be syntactically or semantically malformed, which is usually detected at submission time (either automatically by a validation script or manually by the organizers). In case of errors, a run may need to be repeated in its entirety without a restore point being available. Run file submissions also require the most resources from the participants (e.g., hardware or costs for

[2] https://trec.usersim.ai/

[3] Here, a "test set" contains a topic description and an initial query that starts the dialogue.

Table 1: Comparison of advantages for participants and organizers of shared tasks in conversational search with run file submissions (participants run both system and simulator), API-based submissions (participant runs system and organizer the user simulator, as enabled by Sim.API), or software submissions (organizer runs both system and simulator).

	Submission format		
	Run file	**API**	**Software**
For participants			
Submission requires little technical skills	yes	yes	no
Early error detection	no	yes	no
System development task-independent	yes	yes	no
Manual runs possible	yes	yes	no
System uses organizer resources	no	no	yes
Simulator uses organizer resources	no	yes	yes
For organizers			
Low support & maintenance effort	yes	yes	no
Secret test datasets	no	yes	yes
Can track system	no	no	yes
Can track simulator	no	yes	yes
Reusable software artifacts	no	no	yes
Simulator manipulation-resistant	no	yes	yes

external services like commercial LLMs). Another downside of run file submissions is that the test dataset has to be made public, since it has to be downloaded to a participant's hard drive. This data could be obfuscated through encryption, but this requires considerable effort on the part of the organizers to ensure that participants cannot recover it. Additionally, participants could manipulate the user simulators to get an unfair advantage in the competition.

Software Submissions. Software submission requires participants packaging their system's software and submitting it to the organizers, who then run the software and simulated users on their own hardware. Software can be submitted by either a prepackaged artifact (e.g., Docker image) or source code that organizers can package automatically. Such software artifacts can be executed on external servers or within integrated experiment platforms such as TIRA [16], TIREx [15], or CodaLab [22]. While SimLab [6] represents a more specialized platform for automating user simulation, other platforms also enable interactions between participant systems and real humans such as STELLA Living Lab [7] or EvalAI [27].

In particular, software submissions through dockerized software artifacts have been used by the Touché Retrieval-Augmented Debating (RAD) [18] task organizers to enable conversations between simulated users and participant systems. Apart from their use for simulations in conversational search, software submissions have also been used in recent shared tasks in ad hoc retrieval such as LongEval'25 [9] or the ReNeuIR'25 shared task [8].

As highlighted in Table 1, software submissions require the most technical skills and effort in exchange for control by the organizers. However, participants benefit from the lack of hardware requirements (except during the development process), since participant systems are executed on the hardware of the organizers. One of the key advantages is that a reusable software artifact is created that can be used to reproduce results and to repeat evaluations on different test datasets or with different configurations. Software submissions also allow for secret test datasets and reduce the risk of simulator manipulations by participants due to sandbox execution. The main disadvantage is the substantial increase of technical expertise that is required for participants to develop such reusable artifacts. The high technical hurdle can significantly reduce the number of teams participating in a shared task and inevitably increases the support effort for the organizers, who have to help participants with their submissions: The execution of participant systems on foreign hardware and unseen data can also lead to crashes of the system and errors, which can only be solved in collaboration between organizers and participants. In terms of software stack limitations, participants are flexible but have to adapt their code to support organizer's hardware (e.g., CUDA, PyTorch, TensorFlow).

API-Based Submissions. An API-based submission format requires the organizers to provide a publicly accessible API as a web service through which the participant systems are queried. In the specific case of conversational search evaluation with user simulators, participant systems exchange utterances with the user simulators through the API. In contrast to software submissions, the participant system is executed at the participant's side, positioning API-based submissions between the other two formats. The API provides the functionality for establishing the connection between simulators and systems, recording conversations, monitoring progress, and maintaining run data of participating teams. From the collected data, which contains information about the submitted runs, the organizers can extract information in the quantity and format they deem appropriate for conducting the evaluation.

To the best of our knowledge, the interactive task at TREC iKAT 2025 [4] was the first shared task to adopt an API-based submission procedure. Specifically, our proposed Sim.API middleware was used to implement the submission system.

Similarly to run file submissions, API-based submission systems require a lower level of technical skills from the participants. The required skills are limited to knowledge about the communication (e.g., HTTP) and interaction protocols (i.e., the semantics of endpoints). Participants have the freedom of submitting requests to the API with the software libraries, frameworks, or even programming languages they are most familiar with, reducing the barrier for participation. As the middleware monitors the individual requests, immediate feedback can be given about errors (e.g., an incorrect request), which eliminates the risk of faulty run submissions, because a run is not continued in the event of an error. All these points help to minimize the need for support from the organizers. Encapsulation between systems and simulators also reduces the risk of malicious actions and enables secret test datasets. One disadvantage of this approach, as with the submission of run files, is that participants do not create software artifacts that can be reused by the organizers without additional effort.

3 Sim.API

Sim.API is a middleware technology with a stateful HTTP API that enables participants to receive utterances from simulated users, provide system responses, and thereby generate run submissions without the need for participants to create run files. Figure 1 gives

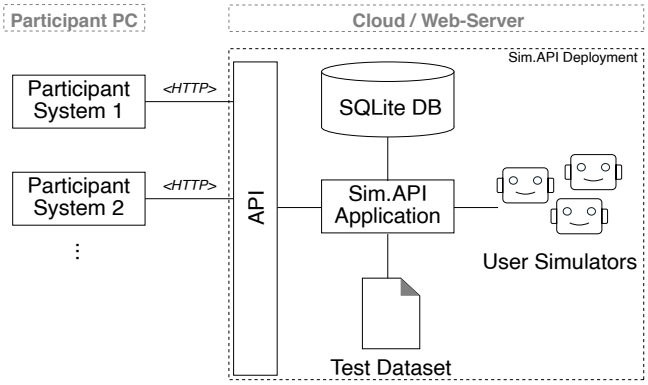


Figure 1: Example deployment of Sim.API. Alternatively, the user simulators can be deployed externally and communicate with the Sim.API-middleware through HTTP.

Table 2: Overview of API endpoints of Sim.API. The endpoints marked with the lock icon (🔒) require administrator privileges.

API Endpoint		Description
POST	`/run/start`	Register a run and receive the first user utterance.
POST	`/run/continue`	Submit a response and receive the next user utterance.
GET	`/run/session`	Get the conversation history of the current session.
GET	`/run/status`	Get information about the status and progress of one of the participants' runs.
POST	`/debug/start`	Register a test run and receive the first user utterance.
POST	`/debug/continue`	Submit a response and receive the next user utterance within a test run.
GET	`/debug/session`	Get the conversation history of the current session in a test run.
GET	`/auth/issue-token` 🔒	Register a new team.
GET	`/auth/verify`	Verify the validity of an access token.
GET	`/budget/check`	Check how much submission budget is left for using debug and run endpoints.
GET	`/run/dump`	Export a run file of a specific run.
GET	`/run/dump-all` 🔒	Export run files of all runs.

an overview of an exemplary deployment of Sim.API for a conversational search shared task. Participants develop conversational search systems on their local hardware. These systems connect through the provided API with the user simulators, which the Sim.API application orchestrates, monitors, and records. All relevant interactions as well as metadata are stored in a relational database. In the following, we detail each functionality behind individual HTTP endpoints that Sim.API offers (overview in Table 2) to participants and organizers of shared tasks.

```
{
  "timestamp":"2025-05-06T10:58:33.400920",
  "run_id":"teamA-llama3-dense-retrieval",
  "topic_id":"1-1",
  "user_id":"f3333f5c-8313-4941-beea-f24de25a583a",
  "utterance":"What universities have CS programs?",
  "history":[
    {"role":"user",
     "content":"What universities have CS programs?"}
  ],
  "last_response_of_session":false,
  "last_response_of_run":false
}
```

Figure 2: Example of an API response object representing a simulated user utterance.

```
{
   "run_id": "teamA-llama3-dense-retrieval",
   "response": "There is University of Amsterdam.",
   "citations": {
     "clueweb22-en0032-04-00208:7": 0.8,
     "clueweb22-en0027-84-11778:0": 0.4,
   },
   <metadata>
}
```

Figure 3: Example of an API request object representing a system response of a participant system.

3.1 Run Submission

The HTTP API offers a simple interface that allows participants to exchange utterances with the available user simulators. A complete run consists of a conversation (search session) between the participant system and a simulated user about each topic in the test dataset. To submit a run, participants use two different HTTP POST endpoints (`/run/start` and `/run/continue`). First, a participant registers a new run by sending run metadata to the `/run/start` route. The metadata for registering a run includes a unique identifier and a brief description of the submitted approach. If the given run identifier is already taken, the run registration is rejected, and the participant receives an error message explaining the issue. In case of successful run registration, participant systems obtain the initial utterance of a simulated user. The participant system uses the initial user message to create a response, which is sent to the `/run/continue` route to obtain the next user utterance.

Figure 2 shows an example of a response from the API, which represents a user utterance. The response contains a timestamp, the run identifier, an identifier for the topic from the test dataset this utterance is about, an identifier for the user simulator that produced the utterance, and the current conversation. The user utterance itself is provided separately in two formats: (1) as a text field and (2) as a list of messages that is suitable for large language model inference. This list contains the entire conversation history, including all exchanged utterances. The response also contains two flags indicating the end of the current session and the entire run.

An exemplary system response to the above user utterance can be seen in Figure 3. Participants have to transmit the identifier of the run they want to progress, the text response, as well as a list of referenced documents and their corresponding relevance scores.

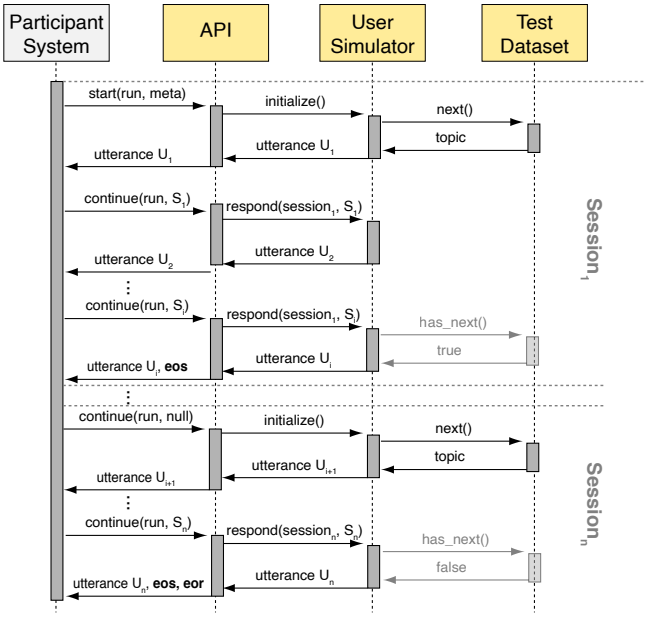


Figure 4: UML sequence diagram that demonstrates the interaction protocol of Sim.API. `eos` **and** `eor` **are flags to indicate the end of a session or of a run. Components in yellow color (▭) indicate a server-side deployment.**

Additionally, organizers can configure metadata that participants must provide to submit a turn.

Figure 4 illustrates the process of an entire run submission. The participant system registers the run, which prompts the user simulator to generate the initial utterance of the session about the first topic in the test dataset. The session is progressed by the participant system by responding to each user utterance by posting generated responses to the `/run/continue` route, receiving the next user utterance in return. The responsibility to end the current session lies with the user simulator, which is indicated with the above-mentioned flag. If the participant system continues after a session was ended, the simulator clears the context and provides the initial utterance of the next session about the next available topic from the test dataset. However, if the user simulator terminates a session and no more topics are available in the test dataset, the API response indicates the end of the current run. Further attempts to continue are rejected with a descriptive error message.

In the current version of Sim.API, shared task organizers can configure the topics, define the order in which the topics will be processed, supply simulator implementations, and assign simulators to individual topics by creating a designated shared task class in the framework with predefined interfaces. Sim.API provides a set of baseline simulator implementations for zero-shot simulations with open-source or commercial large language models. Additional user simulators can be implemented by shared task organizers by simply overriding two methods that produce the initial utterance (`initialize()`) or respond to system responses (`respond()`). To ease the implementation of new simulation approaches, wrapper classes for transformer-based language models hosted on Huggingface[4] or for language models from OpenAI[5] are part of the framework. The default configuration of the transformer-based models are optimized for fast inference by providing CUDA support for parallelizing computations on GPUs and by attention computations with Flash Attention 2 [14]. To reduce hardware requirements, the language models can be loaded with quantized weights.

3.2 Participant Playground

As participants can run into errors during the development of their systems, especially when they have to interact with an unfamiliar API, Sim.API relies on strict input validations and descriptive error messages. In addition, Sim.API offers a "playground" where participants can interact with simulators via the API without their interactions being counted as run submissions. Part of Sim.API are semantically congruent endpoints to the endpoints for submitting runs, provided at different routes (`/debug/start` and `/debug/continue`), which enable participants to mimic a complete run submission with their developed systems. Organizers of the shared task can configure a separate set of topics and simulator implementations that are served behind the playground endpoints to protect their confidentiality and avoid abuse.

3.3 Authentication and Authorization

A stateful interaction involving session tracking requires the client to authenticate itself to the server, so that the server can assign the correct session information to the associated client. This means for Sim.API that participating teams have to authenticate themselves to the API before each interaction. Additionally, the API exposes computation resources that are associated with costs (e.g., energy consumption, external service costs), which shall only be provided to legitimate participants of the shared task.

Sim.API implements a two-level permission system distinguishing the administrator and the participant roles. The administrator is authorized to (de)register participant teams, access all run information, and reset submission budgets (see Sec. 3.4). Credentials (i.e. username and password) of the administrator account are configured as command line parameters of the server run command. These credentials are hashed with the bcrypt hashing function [23] and stored in the internal database. To access endpoints with administrator privileges, organizers can use the basic access authentication method, in which username and password are encoded as a base64-encoded string and provided as HTTP authorization header.

When the administrator registers a team by submitting a request to the `/auth/issue-token` route, a 16 byte hexadecimal token is generated. The base64-encoded version of the token is given to the organizer who distributes it to the respective participants, while the bcrypt-hashed version is stored in the internal database. To authenticate themselves, participants have to provide the token as a bearer token in the HTTP authorization header to operate the API. The token is verified, and on failure, participant requests result in status codes 401 (unauthorized). For convenience, participants can verify if their token is correct by using the `/auth/verify` endpoint.

[4] https://huggingface.co/
[5] https://openai.com/

3.4 Submission Budgets

To limit the access to the computing resources and to avoid unfair distribution of API usage, organizers can configure submission budgets. Separate budget limits can be set for run submissions and for playground access by modifying a configuration file. This configuration file can be manipulated while the service is running which triggers an automatic reconfiguration and eliminates the need to restart the service to change budget limits. The limits can be specified as maximum number of runs or sessions. Participants can look up how much budget they have left by sending a request to the `/budget/check` route.

3.5 Data Management

All relevant data, which includes run data, logging data, and account information is stored in a relational database. Specifically, in our implementation we use SQLite[6] as it is easy to maintain, ACID-compliant, and requires no overhead for deployment, since the database files are written directly to the file system. In-memory objects of Sim.API such as session and run information are synchronized regularly. In case of a server crash, crucial in-memory objects are recovered from the database, which reduces the risk of data loss or inconsistent states. In the event of a server outage, participants will lose only their last turn if they are attempting to submit their response at the same time of the crash. This also means that participants can pause a run before completion, take a break, complete it at a later time, or even start several parallel runs.

Organizers as well as participants have the ability to export run files from the recorded interactions. Of course, participants may only export the run files of their own submitted runs. To this end, participants can make requests to the `/run/dump` route with `run_id` as URL parameter. The output is a line-limited JSON file that complies with the TREC 2025 run file format. The administrator, on the other hand, can export all submitted runs at once with the `/run/dump-all` endpoint.

3.6 Implementation and Deployment

Sim.API is implemented in Python and tested for version 3.12 and 3.13. The server backend is implemented with FastAPI[7] behind a Uvicorn[8] production server. Sim.API can be deployed either from source code or as a prebuilt Docker image. For the former, the code can be built either with Poetry[9] or with the included Dockerfile. To ensure robustness of the implementation, integration tests are performed and are part of the public repository.

3.7 Documentation

The public repository contains a README file with detailed instructions on how to deploy and how to operate the API from an organizer's and a participant's perspective. Additionally, Sim.API provides an automatically generated documentation of all endpoints and their expected input and output objects at the `/docs` route, which is compliant with the OpenAPI 3.1 documentation standard rendered with the Swagger tool.[10]

4 Past and Planned Deployments

Sim.API was successfully deployed as part of the interactive task of TREC iKAT'25 [4]. Five teams participated in the interactive tasks and submitted a total of 16 runs. Additionally, the organizers provided seven baseline runs through our API. In post hoc analyses of the shared task, 51 additional runs have been submitted, which totals 74 run submissions that have been recorded by Sim.API over the span of several months. During that time, no technical issues have come to our attention, which indicates the robustness of the implementation. Throughout the entire duration of iKAT'25, there was only one support request from participants regarding the API. This support request was suggesting a feature that allows participants to rename already submitted runs. This experience illustrates how easy it is for participants to submit their runs, and that the organizers hardly need to assist with the submission process.

Due to this positive experience from the deployment, Sim.API is planned to be used for iKAT'26, not as part of TREC but in the context of the Search-Oriented Conversational AI (SCAI) workshop at SIGIR 2026 [10]. In addition, Sim.API will be adapted to also provide access to conversational systems so that it can be made available for use in the TREC 2026 User Simulation Track.

5 Conclusion

This paper introduced Sim.API, a middleware that enables API-based submissions to support organizers and participants in shared tasks in conversational search with user simulators. We highlighted the limitations of run file and software submissions for conversational search tasks and discussed how Sim.API addresses them. Among other things, our proposed paradigm requires less technical knowledge on the part of participants to submit their runs compared to software submissions, and allows organizers to use secret topic datasets for evaluating participants' systems compared to run file submissions. The practical deployment of Sim.API in the TREC iKAT'25 shared task demonstrated its robustness and ease of use for participants, causing little support effort.

As for future plans for Sim.API, the current version only includes the server-side application, but a client-side API wrapper could help to further reduce the submission complexity for participants. The client-side software could be distributed as a PyPI package, allowing participants to install the package via pip, configure the host URL and access token, and enable them start right away to develop their systems. Furthermore, the ability to configure different shared task setups requires modification of source code. We plan to streamline the task configuration, so that it can be entirely done by modifying configuration files. Finally, the planned deployments in iKAT'26 and TREC 2026 User Simulation track will reveal additional insights into desirable properties and features of Sim.API, which contributes to an iterative development of our approach, so that it becomes an essential tool for shared task organizers in conversational search with evaluations by user simulation.

[6] https://sqlite.org/
[7] https://fastapi.tiangolo.com/
[8] https://uvicorn.dev/
[9] https://python-poetry.org/
[10] https://swagger.io/

Acknowledgments

This research was supported in part by the Informatics Institute (IvI), University of Amsterdam, and in part by the Swiss National Science Foundation (SNSF) grant 215742, entitled "Personality and Conversational Information Access (PACINO)."

The authors acknowledge the peoples of the Woi Wurrung and Boon Wurrung language groups of the eastern Kulin Nation on whose unceded lands ACM SIGIR 2026 was hosted. We pay our respects to their Elders past and present, and extend that respect to all Aboriginal and Torres Strait Islander peoples today and their continuing connection to land, sea, sky, and community.

The authors thank Ian Soborrof for his support in designing and running the human annotation tasks, as well as the NIST assessors for annotating the data. Finally, we thank the TREC iKAT 2025 participants for submitting their work and presenting them in the TREC conference.

References

[1] Jafar Afzali, Aleksander Mark Drzewiecki, Krisztian Balog, and Shuo Zhang. 2023. UserSimCRS: A User Simulation Toolkit for Evaluating Conversational Recommender Systems. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, USA, 1160–1163. doi:10.1145/3539597.3573029

[2] Mohammad Aliannejadi, Zahra Abbasiantaeb, Shubham Chatterjee, Jeffery Dalton, and Leif Azzopardi. 2023. TREC iKAT 2023: The Interactive Knowledge Assistance Track Overview. In *The Thirty-Second Text REtrieval Conference Proceedings (TREC 2023), Gaithersburg, MD, USA, November 14-17, 2023 (NIST Special Publication, Vol. 1328)*, Ian Soboroff and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST), Gaithersburg, USA, 11 pages. https://trec.nist.gov/pubs/trec32/papers/Overview_ikat.pdf

[3] Mohammad Aliannejadi, Zahra Abbasiantaeb, Simon Lupart, Shubham Chatterjee, Jeffrey Dalton, and Leif Azzopardi. 2024. TREC iKAT 2024: The Interactive Knowledge Assistance Track Overview. In *Proceedings of the Thirty-Third Text REtrieval Conference, TREC 2024, Gaithersburg, MD, USA, November 18-22, 2024 (NIST Special Publication, Vol. 1329)*, Ian Soboroff, Hoa Dang, and George Awad (Eds.). National Institute of Standards and Technology (NIST), Gaithersburg, USA, 12 pages. https://trec.nist.gov/pubs/trec33/papers/Overview_ikat.pdf

[4] Mohammad Aliannejadi, Simon Lupart, Marcel Gohsen, Zahra Abbasiantaeb, Nailia Mirzakhmedova, Johannes Kiesel, and Jeffrey Dalton. 2025. TREC iKAT 2025: The Interactive Knowledge Assistance Track Overview. In *Proceedings of the 34th Text REtrieval Conference (TREC 2025) (NIST Special Publication, Vol. 1348)*, Ian Soboroff and George Awad (Eds.). National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA, 15 pages. https://pages.nist.gov/trec-browser/trec34/ikat/proceedings/#trec-ikat-2025-the-interactive-knowledge-assistance-track-overview

[5] Leif Azzopardi, Timo Breuer, Björn Engelmann, Christin Kreutz, Sean MacAvaney, David Maxwell, Andrew Parry, Adam Roegiest, Xi Wang, and Saber Zerhoudi. 2024. SimIIR 3: A Framework for the Simulation of Interactive and Conversational Information Retrieval. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region, SIGIR-AP 2024, Tokyo, Japan, December 9-12, 2024*, Tetsuya Sakai, Emi Ishita, Hiroaki Ohshima, Faegheh Hasibi, Jiaxin Mao, and Joemon M. Jose (Eds.). ACM, New York, USA, 197–202. doi:10.1145/3673791.3698427

[6] Nolwenn Bernard, Sharath Chandra Etagi Suresh, Krisztian Balog, and Chengxiang Zhai. 2025. SimLab: A Platform for Simulation-based Evaluation of Conversational Information Access Systems. doi:10.48550/arXiv.2507.04888

[7] Timo Breuer and Philipp Schaer. 2021. A Living Lab Architecture for Reproducible Shared Task Experimentation. In *Information between Data and Knowledge: Information Science and its Neighbors from Data Science to Digital Humanities - Proceedings of the 16th International Symposium of Information Science, ISI 2021, Regensburg, Germany, March 8-10, 2021*, Christian Wolff and Thomas Schmidt (Eds.). Werner Hülsbusch, Glückstadt, Germany, 348–362. doi:10.5283/EPUB.44953

[8] Sebastian Bruch, Maik Fröbe, Tim Hagen, Franco Maria Nardini, and Martin Potthast. 2025. ReNeuIR at SIGIR 2025: The Fourth Workshop on Reaching Efficiency in Neural Information Retrieval. In *48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2025)*, Nicola Ferro, Maria Maistro, Gabriella Pasi, Omar Alonso, Andrew Trotman, and Suzan Verberne (Eds.). ACM, New York, 4153–4156. doi:10.1145/3726302.3730358

[9] Matteo Cancellieri, Alaa El-Ebshihy, Tobias Fink, Maik Fröbe, Petra Galuscáková, Gabriela González Sáez, Lorraine Goeuriot, David Iommi, Jüri Keller, Petr Knoth, Philippe Mulhem, Florina Piroi, David Pride, and Philipp Schaer. 2025. LongEval at CLEF 2025: Longitudinal Evaluation of IR Systems on Web and Scientific Data. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 16th International Conference of the CLEF Association, CLEF 2025, Madrid, Spain, September 9-12, 2025, Proceedings (Lecture Notes in Computer Science, Vol. 16089)*, Jorge Carrillo-de-Albornoz, Alba García Seco de Herrera, Julio Gonzalo, Laura Plaza, Josiane Mothe, Florina Piroi, Paolo Rosso, Damiano Spina, Guglielmo Faggioli, and Nicola Ferro (Eds.). Springer, 363–387. doi:10.1007/978-3-032-04354-2_20

[10] Philipp Christmann, Roxana Petcu, Sneha Singhania, Mohammad Aliannejadi, Marcel Gohsen, and Svitlana Vakulenko. 2026. 10th Workshop on Search-Oriented Conversational Artificial Intelligence (SCAI 2026). In *49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26)*. ACM, New York. doi:10.1145/3805712.3808648

[11] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2019. TREC CAsT 2019: The Conversational Assistance Track Overview. In *Proceedings of the Twenty-Eigth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13–15, 2019 (NIST Special Publication, Vol. 1250)*, Ellen M. Voorhees and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST), Gaithersburg, USA, 10 pages. https://trec.nist.gov/pubs/trec28/papers/OVERVIEW.CAsT.pdf

[12] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2020. CAsT 2020: The Conversational Assistance Track Overview. In *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020 (NIST Special Publication, Vol. 1266)*, Ellen M. Voorhees and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST), Gaithersburg, USA, 10 pages. https://trec.nist.gov/pubs/trec29/papers/OVERVIEW.C.pdf

[13] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2021. TREC CAsT 2021: The Conversational Assistance Track Overview. In *Proceedings of the Thirtieth Text REtrieval Conference, TREC 2021, online, November 15-19, 2021 (NIST Special Publication, Vol. 500-335)*, Ian Soboroff and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST), Gaithersburg, USA, 7 pages. https://trec.nist.gov/pubs/trec30/papers/Overview-CAsT.pdf

[14] Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. arXiv:2307.08691 [cs.LG] https://arxiv.org/abs/2307.08691

[15] Maik Fröbe, Jan Heinrich Reimer, Sean MacAvaney, Niklas Deckers, Simon Reich, Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2023. The Information Retrieval Experiment Platform. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023)*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, New York, USA, 2826–2836. doi:10.1145/3539618.3591888

[16] Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. 2023. Continuous Integration for Reproducible Shared Tasks with TIRA.io. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III*. Springer-Verlag, Berlin, Heidelberg, 236–241. doi:10.1007/978-3-031-28241-6_20

[17] Theo Härder and Andreas Reuter. 1983. Principles of Transaction-Oriented Database Recovery. *Comput. Surveys* 15, 4 (1983), 287–317. doi:10.1145/289.291

[18] Johannes Kiesel, Çağrı Çöltekin, Marcel Gohsen, Sebastian Heineking, Maximilian Heinrich, Maik Fröbe, Tim Hagen, Mohammad Aliannejadi, Sharat Anand, Tomaž Erjavec, Matthias Hagen, Matyáš Kopp, Nikola Ljubešić, Katja Meden, Nailia Mirzakhmedova, Vaidas Morkevičius, Harrisen Scells, Moritz Wolter, Ines Zelch, Martin Potthast, and Benno Stein. 2025. Overview of Touché 2025: Argumentation Systems. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 16th International Conference of the CLEF Association (CLEF 2025) (Lecture Notes in Computer Science)*, Jorge Carrillo de Albornoz, Julio Gonzalo, Laura Plaza, Alba García Seco de Herrera, Josiane Mothe, Florina Piroi, Paolo Rosso, Damiano Spina, Guglielmo Faggioli, and Nicola Ferro (Eds.). Springer, Berlin Heidelberg New York, 486–508.

[19] Johannes Kiesel, Marcel Gohsen, Nailia Mirzakhmedova, Matthias Hagen, and Benno Stein. 2024. Who Will Evaluate the Evaluators? Exploring the Gen-IR User Simulation Space. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 15th International Conference of the CLEF Association (CLEF 2024) (Lecture Notes in Computer Science, Vol. 14958)*, Lorraine Goeuriot, Philippe Mulhem, Georges Quénot, Didier Schwab, Giorgio Maria Di Nunzio, Laure Soulier, Petra Galuscakova, Alba García Seco de Herrera, Guglielmo Faggioli, and Nicola Ferro (Eds.). Springer, Berlin Heidelberg New York, 166–171. doi:10.1007/978-3-031-71736-9_11

[20] David Maxwell and Leif Azzopardi. 2016. Simulating Interactive Information Retrieval: SimIIR: A Framework for the Simulation of Interaction. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, Raffaele Perego, Fabrizio Sebastiani, Javed A. Aslam, Ian Ruthven, and Justin Zobel (Eds.). ACM, New York, USA, 1141–1144. doi:10.1145/2911451.2911469

[21] Paul Owoicho, Jeff Dalton, Mohammad Aliannejadi, Leif Azzopardi, Johanne R. Trippas, and Svitlana Vakulenko. 2022. TREC CAsT 2022: Going Beyond User Ask and System Retrieve with Initiative and Response Generation. In *Proceedings*

of the Thirty-First Text REtrieval Conference, TREC 2022, online, November 15-19, 2022 (NIST Special Publication, Vol. 500-338), Ian Soboroff and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST), Gaithersburg, USA, 12 pages. https://trec.nist.gov/pubs/trec31/papers/Overview_cast.pdf

[22] Adrien Pavao, Isabelle Guyon, Anne-Catherine Letournel, Dinh-Tuan Tran, Xavier Baró, Hugo Jair Escalante, Sergio Escalera, Tyler Thomas, and Zhen Xu. 2023. CodaLab Competitions: An Open Source Platform to Organize Scientific Challenges. *Journal of Machine Learning Research* 24 (2023), 198:1–198:6. https://jmlr.org/papers/v24/21-1436.html

[23] Niels Provos and David Mazières. 1999. A Future-Adaptable Password Scheme. In *Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference, June 6-11, 1999, Monterey, California, USA*. USENIX, Berkely, USA, 81–91. http://www.usenix.org/events/usenix99/provos.html

[24] Philipp Schaer, Christin Katharina Kreutz, Krisztian Balog, Timo Breuer, Andreas Kruff, Mohammad Aliannejadi, Christine Bauer, Nolwenn Bernard, Nicola Ferro, Marcel Gohsen, Nurul Lubis, and Saber Zerhoudi. 2025. Report on the Second Workshop on Simulations for Information Access (Sim4IA 2025) at SIGIR 2025. *SIGIR Forum* 59, 2 (Dec. 2025), 15 pages. doi:10.1145/3799914.3799927

[25] Elizabeth Silliman, Julien Boudet, Kelsey Robinson, Desirae Oppong, and Nilay Shah. 2025. New front door to the internet: Winning in the age of AI search. https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/new-front-door-to-the-internet-winning-in-the-age-of-ai-search. *McKinsey's Growth, Marketing & Sales Practice* (2025).

[26] Svitlana Vakulenko, Johannes Kiesel, and Maik Fröbe. 2022. SCAI-QReCC Shared Task on Conversational Question Answering. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference, LREC 2022, Marseille, France, 20-25 June 2022*, Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis (Eds.). European Language Resources Association, Paris, France, 4913–4922. https://aclanthology.org/2022.lrec-1.525

[27] Deshraj Yadav, Rishabh Jain, Harsh Agrawal, Prithvijit Chattopadhyay, Taranjeet Singh, Akash Jain, Shiv Baran Singh, Stefan Lee, and Dhruv Batra. 2019. EvalAI: Towards Better Evaluation Systems for AI Agents. http://arxiv.org/abs/1902.03570

[28] Saber Zerhoudi, Sebastian Günther, Kim Plassmeier, Timo Borst, Christin Seifert, Matthias Hagen, and Michael Granitzer. 2022. The SimIIR 2.0 Framework: User Types, Markov Model-Based Interaction Simulation, and Advanced Query Generation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, New York, USA, 4661–4666. doi:10.1145/3511808.3557711