A Simulation-Based Evaluation Framework for Interactive AI Systems and Its Application

Maeda F. Hanafi*, Yannis Katsis*, Martín Santillán Cooper*, Yunyao Li

IBM Research AI

{maeda.hanafi, yannis.katsis, msantillancooper}@ibm.com, yunyaoli@us.ibm.com

Abstract

Interactive AI (IAI) systems are increasingly popular as the human-centered AI design paradigm is gaining strong traction. However, evaluating IAI systems, a key step in building such systems, is particularly challenging, as their output highly depends on the performed user actions. Developers often have to rely on limited and mostly qualitative data from ad-hoc user testing to assess and improve their systems. In this paper, we present *InteractEva*; a systematic evaluation framework for IAI systems. We also describe how we have applied InteractEva to evaluate a commercial IAI system, leading to both quality improvements and better data-driven design decisions.

Introduction

Classical AI systems are based on a two-step development workflow, where developers create an AI model based on labels provided by Subject Matter Experts (SMEs), which is then deployed and made available for SMEs to use. As a result, SMEs are not directly involved in the model building process, with their feedback incorporated only after lengthy discussions with developers or other mediators (Amershi et al. 2014). To empower users and build better AI systems, the community has looked into building AI systems with humans-in-the-loop. Particularly popular have been *interactive ML/AI (IAI)* systems, which continuously interact with SMEs and incorporate their feedback to create ever-improving versions of the underlying AI models (Fails and Olsen Jr 2003; Amershi et al. 2014).

While a lot of work focuses on the development of IAI systems, there are many unaddressed challenges when it comes to their *evaluation*. Since the resulting AI model depends on the performed user actions, *how can developers of such systems understand and track their performance accurately and efficiently?* A common technique is to drive evaluation from user testing. SMEs interact with the system to identify and report suboptimal cases, which are then replicated and debugged by developers. While user testing is very valuable, relying solely on it may lead to an ad-hoc whack-a-mole approach towards model improvement that is based only on *limited* evidence of mostly *qualitative* nature.

In this paper, we present *InteractEva*; a novel evaluation framework for IAI systems tailored towards providing *data-driven, quantitative* guidance in the development of IAI systems. InteractEva leverages a *user simulation engine* to simulate a large number of user interactions, automatically collects *quantitative* performance data, and generates visualizations and evaluation reports allowing developers to work with designers and other stakeholders and make data-driven decisions over the development of an IAI system.

Our work makes the following contributions:

- An analysis of the *challenges* of IAI system evaluation and the *desiderata* for an evaluation framework.
- A *novel evaluation framework* for IAI systems. The framework combines a simulation-based backend to automatically test the system against different use cases and user interactions at scale and an interactive frontend, allowing developers to perform quantitative evaluation tasks, including acquiring a performance overview, performing error analysis, and conducting what-if studies.
- A description of how the evaluation framework was used in a *real-life industrial setting* to improve Pattern Induction; a commercial IAI text extraction system.

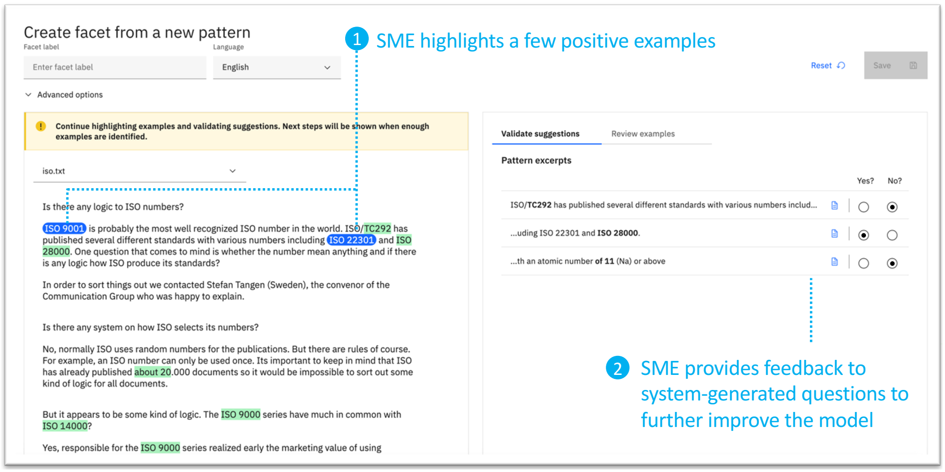
Note that this work focuses on evaluating the backend of an IAI system and its ability to consistently learn high-quality models, which falls under the class of *algorithm-centered* evaluation approaches (Boukhelifa, Bezerianos, and Lutton 2018). IAI systems can also be evaluated on user experience, leading to *human-centered* evaluations (Sperrle et al. 2021), which however are outside the scope of this work. For additional discussions on related work, please refer to the corresponding section at the end of the paper.

Tested IAI System: Pattern Induction

To exemplify our IAI evaluation framework, we will use Pattern Induction; a commercial IAI Natural Language Processing (NLP) extraction system, which is currently available in Beta as part of IBM Watson® Discovery [1]. Given a set of documents, Pattern Induction allows SMEs to interactively extract text mentions that follow recurring patterns. Examples of patterns include ISO numbers (i.e., 'ISO' followed by a number), percentage of crimes by type (i.e., percentage

*These authors contributed equally.

[1] https://www.ibm.com/cloud/watson-discovery


1 SME highlights a few positive examples
2 SME provides feedback to system-generated questions to further improve the model
Reset
Save

Create facet from a new pattern

Facet label: Enter facet label

Language: English

Advanced options

Continue highlighting examples and validating suggestions. Next steps will be shown when enough examples are identified.

iso.txt

Is there any logic to ISO numbers?

ISO 9001 is probably the most well recognized ISO number in the world. ISO/TC292 has published several different standards with various numbers including ISO 22301 and ISO 28000. One question that comes to mind is whether the number mean anything and if there is any logic how ISO produce its standards?

In order to sort things out we contacted Stefan Tangen (Sweden), the convenor of the Communication Group who was happy to explain.

Is there any system on how ISO selects its numbers?

No, normally ISO uses random numbers for the publications. But there are rules of course. For example, an ISO number can only be used once. Its important to keep in mind that ISO has already published about 20.000 documents so it would be impossible to sort out some kind of logic for all documents.

But it appears to be some kind of logic. The ISO 9000 series have much in common with ISO 14000?

Yes, responsible for the ISO 9000 series realized early the marketing value of using

Validate suggestions | Review examples

Pattern excerpts

	Yes?	No?
ISO/**TC292** has published several different standards with various numbers includ...	○	●
...uding ISO 22301 and **ISO 28000**.	●	○
...th an atomic number **of 11** (Na) or above	○	●

Figure 1: Screenshot of Pattern Induction annotated with the types of supported user actions

followed by phrases of the form ‘crimes against property’ or ‘crimes against persons’), COVID-19 cases by country (i.e., country name followed by number), and others.

User actions. To allow SMEs to create an extraction model without requiring a large amount of annotations, Pattern Induction employs an IAI approach. In particular, SMEs can interactively build a model by carrying out two types of *user actions* through the system's UI (see Figure 1):

- *Highlight a small number of positive examples* of mentions to be extracted through the document pane. In Figure 1, an SME willing to extract ISO numbers from her documents has provided two examples ISO 9001, ISO 22301, shown in blue. From these, the system's backend learns a first version of the extraction model and uses active learning to select representative examples and ask for feedback. This leads to another type of action:
- *Provide boolean feedback to system-generated questions*. Figure 1 (right-most pane) shows three examples the system asks the user to verify. Feedback is used by the system to further improve the underlying model.

This loop is repeated until the SME is satisfied with the extraction results (highlighted in green on the document pane and also depicted on a separate review pane, not shown here for space reasons).

Learning algorithm. Throughout this loop, user actions are fed into Pattern Induction's backend, which uses a rule induction algorithm to learn *extraction rules* capturing the desired patterns. Rules in Pattern Induction consist of a sequence of *primitives*, each capturing one or more tokens of an extraction. The following primitive types are supported:

- *regular expressions* – e.g., $\boxed{\text{[A-Za-z]+}}$
- *prebuilt extractors* for standard entities – e.g., $\boxed{\text{IntegerNumber}}$ and $\boxed{\text{Organization}}$
- *token gaps* – e.g., $\boxed{\text{Token gap: 3}}$ (skip over 0 to 3 tokens)
- *literals & dictionaries* – e.g., $\boxed{\text{ISO}}$ & $\boxed{\text{[ISO, IEC, TC]}}$.

For instance, the positive examples of ISO numbers provided above can be captured by the rule $\boxed{\text{ISO}}$ $\boxed{\text{[0-9]+}}$, looking for the string ‘ISO‘ followed by a number. For a comprehensive review of Pattern Induction please refer to SEER (Hanafi et al. 2017), on which the system is based.

Evaluation Challenges & Desiderata

Once an initial version of Pattern Induction was developed, we worked together with other stakeholders (including researchers, designers, and engineers) to understand the performance of the system and improve it. We focused on the backend algorithm trying to answer the following question: *Can the system learn good extraction models for different use cases and if not, what needs to be improved?*

Challenges. To this end, we asked SMEs to interact with the end-to-end system and report cases where the extraction results were not satisfactory. While this successfully revealed some issues, we found the process to not be sufficient for our development workflow for three reasons: First, it is labor-intensive requiring significant effort by SMEs to carry out and document the issues (as they have to also describe the sequence of performed actions). As a result, we received only *limited feedback*, covering a relatively small set of user interactions. Second, the feedback was mostly *qualitative*. While user reports described cases where the learning algorithm was underperforming, we did not have a way to quantify the performance of the learned model and track its performance over time. Third, conducting *error analysis* of the reported issues to identify their root causes and inform model improvement efforts also required substantial work. Due to the interactive nature of the system, developers had to manually write test cases replicating the reported sequence of user actions, in order to then inspect the internal state of the model (i.e., the learned rules) and debug it. As a result, we were left with *limited*, *anecdotal* evidence of *qualitative* nature that required significant time to act on.

Desiderata. Supporting our development workflow required a systematic evaluation framework that would allow

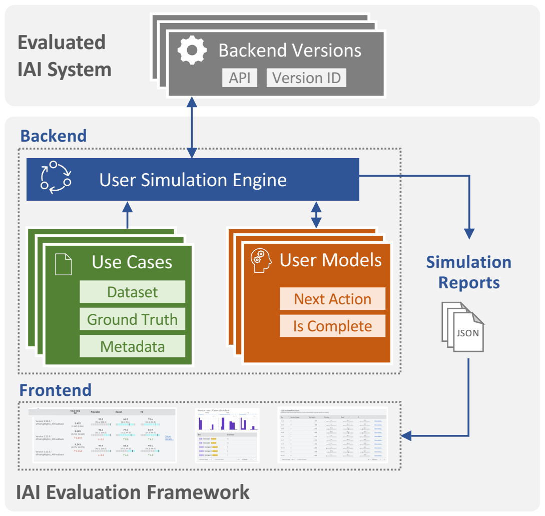


Figure 2: InteractEva's architecture

the development team to try several usage scenarios *at scale* and get a *quantitative* view of the performance of the resulting model. Based on our observations above, as well as interactions with other stakeholders, we identified three main evaluation tasks that such a framework should support:

- *Understand overall AI model performance:* At its most basic form, it should allow developers to quickly understand the performance of the learned model on a variety of user interactions and datasets. The performance should be quantified, so that developers can get an objective view of how the system performs in different scenarios and identify scenarios for which more work is needed.
- *Conduct error analysis:* Once scenarios where the system underperforms are identified, the framework should enable developers to conduct error analysis to identify the underlying issues. To this end, the framework should expose information about the internal state of the learned model (e.g., the set of learned extraction rules).
- *Perform What-If analysis:* Last but not least, the evaluation framework should allow developers to perform comparative What-If analyses to enable data-driven decisions. This may involve comparing different backend versions (e.g., to check how a change to the backend affects performance and avoid regressions) or comparing the result of different user interactions to make UX recommendations (e.g., compare the system performance when a user provides 2 vs 4 positive examples to understand how many positive examples users should be encouraged to provide).

Evaluation Framework Overview

Figure 2 depicts the architecture of *InteractEva*; the resulting evaluation framework. To enable the evaluation of an IAI system at scale, InteractEva is designed around a user simulation engine. This engine (running on the framework's *backend*) is responsible for simulating user actions and feeding them into the tested IAI system, which then learns a respective AI model. The predictions of the AI model (e.g., the extractions in our example) are then retrieved by the framework and evaluated against provided ground truth using standard evaluation metrics (such as Precision, Recall, and F1) to get quantifiable measures of the model's performance. These performance results, together with additional information about the simulations, are passed into the framework's *frontend*, which allows developers to inspect them in a variety of ways to perform the aforementioned evaluation tasks. The frontend-backend interaction happens through JSON files, designed to facilitate the reuse of the frontend for the evaluation of other IAI systems, as we will see later. We next discuss the backend and frontend in detail. In the following, *user* or *SME* refers to the end user of Pattern Induction and *developer* refers to the person using InteractEva to assess the performance of Pattern Induction.

Evaluation Framework Backend

InteractEva's backend is structured around a *user simulation engine* that can simulate large numbers of user interactions and evaluate the performance of the resulting models. As shown in Figure 2, the simulation engine interacts with three components, designed to be flexible and capture the requirements of different IAI systems:

- *Use cases:* Acquiring a good understanding of the performance of an IAI system requires evaluating it on a variety of datasets/tasks. To this end, InteractEva can be supplied with a set of *use cases*. Each use case corresponds to a specific task on a particular dataset (e.g., extracting ISO numbers from ISO reference documents) and consists of the following components: the *dataset* (e.g., input documents), the *ground truth* (e.g., text and span information of the ISO numbers to be extracted), and *metadata* providing additional information about the use case (e.g., description of the task and representative examples). To evaluate Pattern Induction, we created several use cases of varying difficulty (see Table 1 for a subset thereof).
- *User models:* While use cases capture the multitude of tasks on which the IAI system should be tested, the framework should also capture the variety of potential user interactions. This is accomplished by user models. A user model represents a class of potential user interactions, expressed as a sequence of user actions of specific types. For instance, for Pattern Induction, a user model m_1 may represent SMEs that first highlight two positive examples and then provide feedback to all system questions. Similarly, another user model m_2 may represent SMEs that start with *four* positive examples before providing feedback. We found that having several user models designed to test the effect of particular user behaviours can be beneficial for what-if studies. For instance, user models m_1 and m_2 can aid in understanding how performance changes under varying number of positive examples. Implementation-wise, a user model exposes two functions; one returning the next user action (e.g., next positive example) and another indicating whether the current user interaction is complete. Note that user models typically employ random number gen-

Use Case	Description	Representative Examples
COVID-19 cases by country	Extract country names followed by number of COVID-19 cases in parentheses.	Spain (239 932) Malta (620)
Cups multiple forms	Extract amounts in cups.	2 cup 1/4 cup 1 1/2 cup
Earnings time period multiple forms	Extract quarterly time periods incl. year appearing in the beginning or end of the extraction.	2014 First-Quarter fourth-quarter of 2013
ISO numbers multiple forms	Extract prefix containing a combination of "ISO", "TC", and "IEC" followed by an integer.	ISO 639 TC 292 ISO/IEC 40180 ISO/TC 28

Table 1: Subset of information extraction use cases used to evaluate Pattern Induction

erators to simulate multiple user interactions of the same type. For instance, user model m_1 randomly selects from the ground truth two examples to highlight, thus generating a different user interaction upon each invocation.

- *Tested IAI system's backend:* Finally, the simulation engine interacts with the API of the tested IAI system's backend to perform the simulation. Through the API the simulation engine (a) submits the user actions instructed by the respective user model and (b) retrieves the predictions of the learned AI model with their explanations (if available). Model explanations, which are leveraged in error analysis, as we will see next, can be either native explanations of white-box models or explanations created through explainability techniques for black-box models (Xu et al. 2019; Danilevsky et al. 2020). For instance, in Pattern Induction explanations take the form of rules. Note that InteractEva can interface with different versions of the IAI system, allowing experimentation with different backend algorithms or comparison of different backend versions over time.

Running the simulation. For a given backend version B and user model M, the simulation engine iterates over all use cases U. For each use case it queries the user model k times to generate k user simulations, referred to next as *runs* [2]. For each run, it executes the simulation, retrieves the predictions of the learned model by the IAI system and evaluates it against the groundtruth using standard evaluation metrics (e.g., Precision, Recall, and F1). The results of all runs over all use cases are stored in a specially formatted *evaluation report*. Each report captures not only the aggregate evaluation metrics, but detailed information about the simulation, including information about the evaluation results as well as model explanations (e.g., rules) for each individual run. One or more of these reports can then be loaded into InteractEva's frontend, facilitating both inspection of a single (backend version, user model) pair, as well as comparison of different backend versions or user models.

Evaluation Framework Frontend

We next present InteractEva's frontend, explaining in the process how it allowed developers to carry out the evaluation tasks outlined above and improve Pattern Induction. *Note that the presented evaluation results correspond to intermediate development versions and do not necessarily reflect the performance of the commercial product.*

Understanding the Overall AI Model Performance

When invoked with a simulation report for a given version b_1 of Pattern Induction and user model m_1, the frontend shows a tabular summary of model performance across all use cases (Figure 3 (1)). Each cell displays the precision, recall, or F1 of the learned model for a particular use case over all simulated runs. The performance visualization captures both the average performance of the model (black bar), as well as the range of min/max performance observed during the simulation (light blue bars). This helps developers identify not only cases where the IAI system consistently underperforms, but also *long-tail edge cases*, which are especially important for improving AI models (Bornstein and Casado 2020). Through this summary, developers can assess overall performance and decide where to focus their analysis efforts. For example, they can see that the use case *Cups multiple forms* has low F1 and needs further investigation.

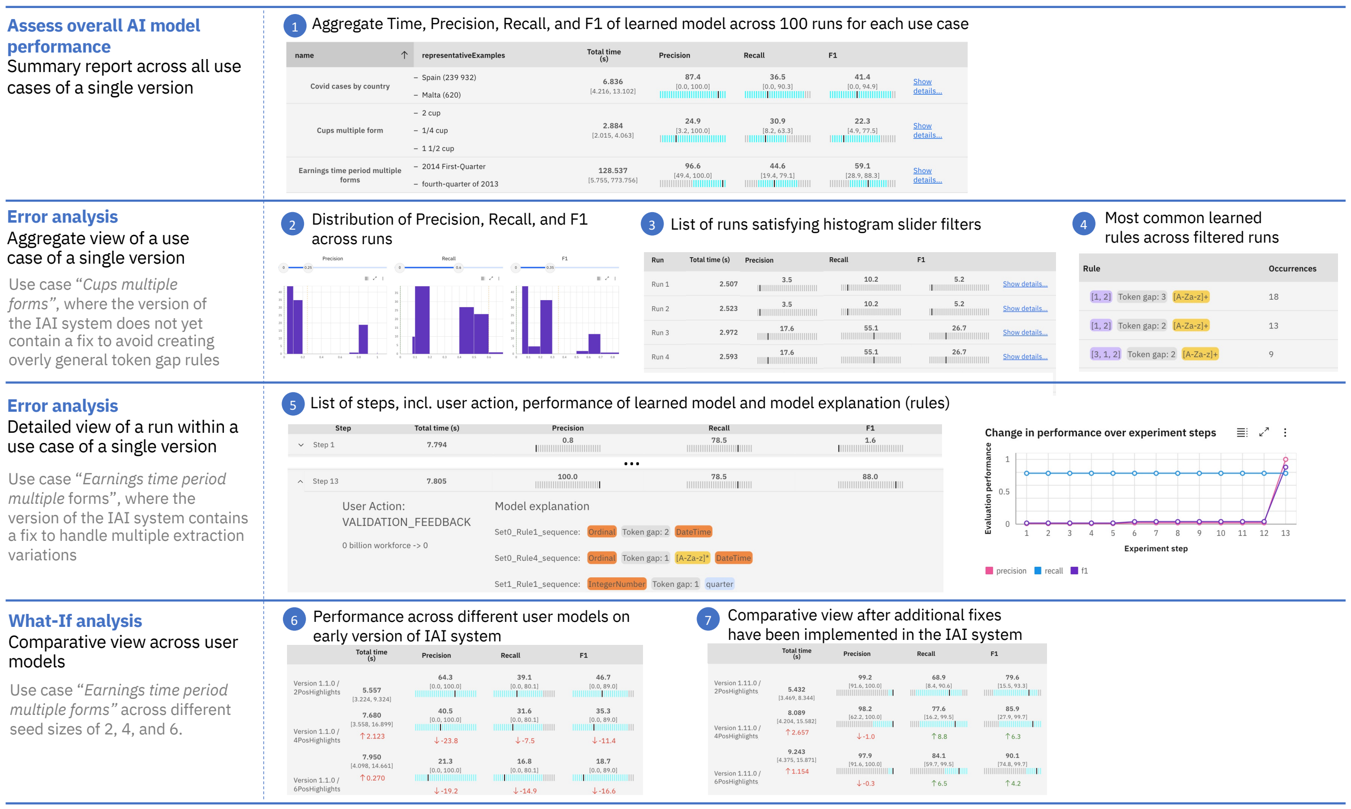
Conducting Error Analysis

Developers can then drill down into specific use cases to identify issues in the IAI system and their root causes. InteractEva supports the following error analysis tasks:

Identify patterns across runs. When inspecting a use case, the framework offers an overview of the model's performance across all simulated runs through the following visualizations: (a) a distribution of evaluation results (e.g., precision, recall, F1) across runs in the form of histograms (Figure 3 (2)), (b) a list showing the performance for each run (Figure 3 (3)), and (c) a summary of the rules learned during the runs and their respective frequency (Figure 3 (4)).

This information can help developers identify error patterns across runs. We next demonstrate this by presenting a real-life error analysis result and resulting fix from the Pattern Induction development process. While inspecting the *Cups multiple forms* use case and studying the histograms, the developer notices that the distributions for precision, recall, and F1 scores skew to the left and merit more inspection (Figure 3 (2)). Using the histogram sliders, she filters the run list and aggregated rule views to keep only low-scoring runs.

[2]For our analysis, we utilized $k = 100$ runs



Assess overall AI model performance
Summary report across all use cases of a single version

1 Aggregate Time, Precision, Recall, and F1 of learned model across 100 runs for each use case

name	representativeExamples	Total time (s)	Precision	Recall	F1	
Covid cases by country	– Spain (239 932) – Malta (620)	6.836 [4.216, 13.102]	87.4 [0.0, 100.0]	36.5 [0.0, 90.3]	41.4 [0.0, 94.9]	Show details...
Cups multiple form	– 2 cup – 1/4 cup – 1 1/2 cup	2.884 [2.015, 4.063]	24.9 [3.2, 100.0]	30.9 [8.2, 63.3]	22.3 [4.9, 77.5]	Show details...
Earnings time period multiple forms	– 2014 First-Quarter – fourth-quarter of 2013	128.537 [5.755, 773.756]	96.6 [49.4, 100.0]	44.6 [19.4, 79.1]	59.1 [28.9, 88.3]	Show details...

Error analysis
Aggregate view of a use case of a single version

Use case *"Cups multiple forms"*, where the version of the IAI system does not yet contain a fix to avoid creating overly general token gap rules

2 Distribution of Precision, Recall, and F1 across runs

3 List of runs satisfying histogram slider filters

Run	Total time (s)	Precision	Recall	F1	
Run 1	2.507	3.5	10.2	5.2	Show details...
Run 2	2.523	3.5	10.2	5.2	Show details...
Run 3	2.972	17.6	55.1	26.7	Show details...
Run 4	2.593	17.6	55.1	26.7	Show details...

4 Most common learned rules across filtered runs

Rule	Occurrences
[1, 2] Token gap: 3 [A-Za-z]+	18
[1, 2] Token gap: 2 [A-Za-z]+	13
[3, 1, 2] Token gap: 2 [A-Za-z]+	9

Error analysis
Detailed view of a run within a use case of a single version

Use case *"Earnings time period multiple* forms", where the version of the IAI system contains a fix to handle multiple extraction variations

5 List of steps, incl. user action, performance of learned model and model explanation (rules)

Step	Total time (s)	Precision	Recall	F1
Step 1	7.794	0.8	78.5	1.6
...				
Step 13	7.805	100.0	78.5	88.0

User Action:
VALIDATION_FEEDBACK

0 billion workforce -> 0

Model explanation

Set0_Rule1_sequence: Ordinal Token gap: 2 DateTime

Set0_Rule4_sequence: Ordinal Token gap: 1 [A-Za-z]* DateTime

Set1_Rule1_sequence: IntegerNumber Token gap: 1 quarter

What-If analysis
Comparative view across user models

Use case *"Earnings time period multiple forms"* across different seed sizes of 2, 4, and 6.

6 Performance across different user models on early version of IAI system

	Total time (s)	Precision	Recall	F1
Version 1.1.0 / 2PosHighlights	5.557 [3.224, 9.324]	64.3 [0.0, 100.0]	39.1 [0.0, 80.1]	46.7 [0.0, 89.0]
Version 1.1.0 / 4PosHighlights	7.680 [3.558, 16.899] ↑2.123	40.5 [0.0, 100.0] ↓-23.8	31.6 [0.0, 80.1] ↓-7.5	35.3 [0.0, 89.0] ↓-11.4
Version 1.1.0 / 6PosHighlights	7.950 [4.098, 14.661] ↑0.270	21.3 [0.0, 100.0] ↓-19.2	16.8 [0.0, 80.1] ↓-14.9	18.7 [0.0, 89.0] ↓-16.6

7 Comparative view after additional fixes have been implemented in the IAI system

	Total time (s)	Precision	Recall	F1
Version 1.11.0 / 2PosHighlights	5.432 [3.469, 8.344]	99.2 [91.6, 100.0]	68.9 [8.4, 90.6]	79.6 [15.5, 93.3]
Version 1.11.0 / 4PosHighlights	8.089 [4.204, 15.582] ↑2.657	98.2 [62.2, 100.0] ↓-1.0	77.6 [16.2, 99.5] ↑8.8	85.9 [27.9, 99.7] ↑6.3
Version 1.11.0 / 6PosHighlights	9.243 [4.375, 15.871] ↑1.154	97.9 [91.6, 100.0] ↓-0.3	84.1 [59.7, 99.5] ↑6.5	90.1 [74.8, 99.7] ↑4.2

Figure 3: Illustration of InteractEva's core frontend features

The resulting aggregate rule view shows that across low scoring runs the most commonly learned rules contain token gaps (which act as wildcards) (Figure 3 (4)). Such rules are *generic* and lead to many incorrect extractions. Based on this insight, the development team introduces optional regular expressions into the learner to prevent such over-generalizations. As a result of the fix, instead of generating the generic rule [1,2] Token gap: 3 [A-Za-z]+, Pattern Induction now generates the more specific rule IntegerNumber [0-9]* [/]* [0-9]* [A-Za-z]+, leading to better performance on both this use case and beyond.

Inspect individual runs. In addition to acquiring an overview of performance across runs, InteractEva also enables developers to drill down and investigate individual runs (Figure 3 (5)). Developers can see a detailed log of the run, including the user action performed at each step, the resulting model performance, and the model explanation (in Pattern Induction's case, the rules). This allows them to inspect and debug runs directly through InteractEva, without having to manually write test cases to replicate the run.

During the development process this view is typically used to both analyze errors and verify fixes. For instance, Figure 3 (5) shows the details of a run on the *Earnings time period multiple forms* use case after a fix had been implemented in Pattern Induction to capture variations in the provided examples. The fix addressed cases where examples followed different patterns and required multiple rules to be covered. With the detailed run view, developers confirmed that the revised algorithm indeed learns a rule for each variation: (a) IntegerNumber Token gap: 1 quarter (capturing examples, such as 2013 second quarter, with the year in front) and (b) Ordinal Token gap: 2 DateTime (for examples, such as third-quarter 2016, with the year at the end).

Performing What-If Analysis

In addition to the inspection of a single simulation report, InteractEva also allows developers to load multiple reports and compare performance across user models or backend versions. This enables them to perform what-if studies in order to test different hypotheses and make data-driven decisions.

Compare performance across user models. For instance, Figure 3 (6) shows the performance of an early version of Pattern Induction on the *Earnings time period multiple forms* use case across three user models, simulating users that start by providing a seed of 2, 4, and 6 positive examples, respectively, before giving feedback to the system-generated questions. Conceptually, more positive examples should improve the performance of the learned model. However, the three user models showed that performance dropped with increasing number of positive examples. This uncovered a deficiency in the backend algorithm for use cases where the examples exhibited higher variability, such as in the use case above, leading to no rules be-

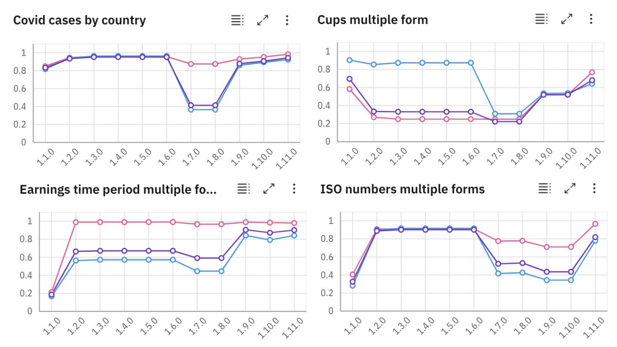


Figure 4: Performance comparison across different backend versions: Average precision (pink), recall (blue), and F1 (purple) over 100 runs each seeded with 6 positive examples.

ing learned. After implementing a fix, the performance of the system improves with increasing number of examples, as verified by the comparison view shown in Figure 3 (7)).

Compare performance across backend versions. Finally, Figure 4 shows charts generated by the frontend comparing the average performance of the system for different use cases over consecutive backend versions. These views proved indispensable in the development process, as they allowed the development team to quantitatively track performance of the system over time and identify and address regressions early on; something that was not possible before the deployment of the evaluation framework.

Impact to Pattern Induction

Once developed, InteractEva was continuously used during the development of Pattern Induction to evaluate the system, identify issues, and inform fixes. At the time of writing, InteractEva has been used for 5 months, during which developers iteratively generated 14 intermediate versions of Pattern Induction. As a result of the functionality outlined above, InteractEva had a multi-faceted impact on Pattern Induction's development and deployment process:

1. it enabled developers to effectively uncover several minor and major issues before deployment,
2. it significantly accelerated debugging efforts with its integrated debugging functionality,
3. it offered a quantitative view of the system's performance, improving in the process the confidence of all stakeholders in the system's abilities,
4. it allowed regressions to be identified early on by enabling tracking of system performance over time, and
5. it enabled the development team to compile guidelines for Pattern Induction users (such as recommendations for the number of provided seed examples) based on the results of what-if analyses.

It is also worth a special mention that InteractEva helped uncover a major class of issues in IAI systems, which are particularly hard to identify solely through user testing: errors that manifest themselves either very rarely or after several user actions. Through its simulation-based architecture, InteractEva helped uncover and rectify several such issues.

Generalizability

While the framework was initially created to evaluate Pattern Induction, generalizability has been a guiding principle since its inception. In particular, the following aspects of the framework can be generalized to other IAI systems:

Reusable frontend. The frontend/backend JSON interface was designed to allow the frontend to be directly reused for the evaluation of other IAI systems. By reading the displayed information (such as user model names, metrics, etc.) directly from the JSON file, the frontend can be used with different evaluation backends that support IAI systems other than Pattern Induction (which may expose different user models, evaluation metrics, or model explanations) [3].

General backend architecture. While the backend will have to be adapted to the tested IAI system (with its own API and ground truth format and its own requirements for user models), we believe that the backend architecture including its modules and interfaces (Figure 2) can be leveraged to develop a backend for another IAI system. Testing this hypothesis and further improving the reusability of the backend will be part of our future work.

Related Work

Evaluation of interactive machine learning. Several works have looked into evaluating IAI systems (see (Boukhelifa, Bezerianos, and Lutton 2018; Sperrle et al. 2021) for recent surveys). These range from human-centered evaluations (focusing on user experience) to algorithm-centered evaluations (studying the robustness of the underlying algorithms). Our work falls under the latter category, but goes beyond existing work by leveraging simulations to build an end-to-end simulation-based evaluation framework that can be generalized to other IAI systems (which to the best of our knowledge is the first of its kind).

User simulation. User simulations have been extensively studied in the context of dialogue systems, where they were used for training (Schatzmann et al. 2007; Kreyssig et al. 2018) or evaluation (Scheffler and Young 2001; Jung et al. 2009; Crook and Marin 2017; Zhang and Balog 2020) purposes. In both cases, the focus has been on generating simulators for the specific task that closely resemble real users. In contrast, our work focuses on an end-to-end evaluation framework (not simply a simulation engine) for general IAI systems (beyond dialogue systems), where one can plug in their own user simulator(s) (which we call *user model(s)*).

Debugging and interpreting AI models. Finally, many works have looked into helping humans understand AI models. These include debugging systems for AI models, such as ModelTracker (Amershi et al. 2015), MLDebugger

[3]The only exception are the aggregate rule views (Figure 3 (4)), which currently require interpretation of the rules and may have to be adapted for other types of model explanations.

(Lourenço, Freire, and Shasha 2019), and ActiVis (Kahng et al. 2018), as well as works on explainable AI (Gilpin et al. 2018; Lipton 2018). These works focused on general AI models without considering how these models were generated. In contrast, our work focuses on models learned by IAI systems, where the resulting model depends on the sequence of user actions. This creates the need for a simulation-based backend and a frontend where simulation and user-related concepts become first-class citizens. However, the aforementioned works can still aid in debugging models learned during individual simulation runs. This is why InteractEva exposes model explanations as part of its simulation reports.

Conclusion

In this work, we introduced InteractEva - a novel evaluation framework for IAI systems - and demonstrated its impact on the development of Pattern Induction; a deployed industrial text extraction system. Through its simulation-based architecture, InteractEva allowed developers to evaluate the system at scale and quickly identify and fix several issues that would have been much harder to spot through traditional user testing. It also allowed them to track the performance of the system over time and make data-driven decisions. As part of our future work, we plan to leverage InteractEva's generalizable architecture to support the development and evaluation of additional IAI systems.

References

Amershi, S.; Cakmak, M.; Knox, W. B.; and Kulesza, T. 2014. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine*, 35(4): 105–120.

Amershi, S.; Chickering, M.; Drucker, S. M.; Lee, B.; Simard, P.; and Suh, J. 2015. *ModelTracker: Redesigning Performance Analysis Tools for Machine Learning*, 337–346. New York, NY, USA: Association for Computing Machinery. ISBN 9781450331456.

Bornstein, M.; and Casado, M. 2020. How to improve AI economics by taming the long tail of data. https://venturebeat.com/2020/08/14/how-to-improve-ai-economics-by-taming-the-long-tail-of-data/. Accessed: 2021-09-16.

Boukhelifa, N.; Bezerianos, A.; and Lutton, E. 2018. Evaluation of interactive machine learning systems. In *Human and Machine Learning*, 341–360. Springer.

Crook, P. A.; and Marin, A. 2017. Sequence to Sequence Modeling for User Simulation in Dialog Systems. In *INTERSPEECH*, 1706–1710.

Danilevsky, M.; Qian, K.; Aharonov, R.; Katsis, Y.; Kawas, B.; and Sen, P. 2020. A Survey of the State of Explainable AI for Natural Language Processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 447–459. Suzhou, China: Association for Computational Linguistics.

Fails, J. A.; and Olsen Jr, D. R. 2003. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, 39–45.

Gilpin, L. H.; Bau, D.; Yuan, B. Z.; Bajwa, A.; Specter, M.; and Kagal, L. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, 80–89. IEEE.

Hanafi, M. F.; Abouzied, A.; Chiticariu, L.; and Li, Y. 2017. *SEER: Auto-Generating Information Extraction Rules from User-Specified Examples*, 6672–6682. New York, NY, USA: Association for Computing Machinery. ISBN 9781450346559.

Jung, S.; Lee, C.; Kim, K.; Jeong, M.; and Lee, G. G. 2009. Data-Driven User Simulation for Automated Evaluation of Spoken Dialog Systems. *Comput. Speech Lang.*, 23(4): 479–509.

Kahng, M.; Andrews, P. Y.; Kalro, A.; and Chau, D. H. 2018. ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1): 88–97.

Kreyssig, F.; Casanueva, I.; Budzianowski, P.; and Gašić, M. 2018. Neural User Simulation for Corpus-based Policy Optimisation of Spoken Dialogue Systems. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, 60–69. Melbourne, Australia: Association for Computational Linguistics.

Lipton, Z. C. 2018. The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3): 31–57.

Lourenço, R.; Freire, J.; and Shasha, D. 2019. Debugging Machine Learning Pipelines. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*, DEEM'19. New York, NY, USA: Association for Computing Machinery. ISBN 9781450367974.

Schatzmann, J.; Thomson, B.; Weilhammer, K.; Ye, H.; and Young, S. 2007. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, 149–152. Rochester, New York: Association for Computational Linguistics.

Scheffler, K.; and Young, S. 2001. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *Proc. NAACL Workshop on Adaptation in Dialogue Systems*, 64–70.

Sperrle, F.; El-Assady, M.; Guo, G.; Borgo, R.; Chau, D. H.; Endert, A.; and Keim, D. 2021. A Survey of Human-Centered Evaluations in Human-Centered Machine Learning. *Computer Graphics Forum*, 40(3): 543–567.

Xu, F.; Uszkoreit, H.; Du, Y.; Fan, W.; Zhao, D.; and Zhu, J. 2019. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *CCF international conference on natural language processing and Chinese computing*, 563–574. Springer.

Zhang, S.; and Balog, K. 2020. *Evaluating Conversational Recommender Systems via User Simulation*, 1512–1520. New York, NY, USA: Association for Computing Machinery. ISBN 9781450379984.