Co-audit: tools to help humans double-check AI-generated content

Andrew D. Gordon, Carina Negreanu, José Cambronero, Rasika Chakravarthy, Ian Drosos, Hao Fang, Bhaskar Mitra, Hannah Richardson, Advait Sarkar, Stephanie Simmons, Jack Williams and Ben Zorn

Microsoft Research

Abstract

Users are increasingly being warned to check AI-generated content for correctness. Still, as LLMs (and other generative models) generate more complex output, such as summaries, tables, or code, it becomes harder for the user to audit or evaluate the output for quality or correctness. Hence, we are seeing the emergence of tool-assisted experiences to help the user double-check a piece of AI-generated content. We refer to these as co-audit tools. Co-audit tools complement prompt engineering techniques: one helps the user construct the input prompt, while the other helps them check the output response. As a specific example, this paper describes recent research on co-audit tools for spreadsheet code powered by generative models. We explain why co-audit experiences are essential for any application of generative AI where quality is important and errors are consequential (as is common in spreadsheet computations). We propose a preliminary list of principles for co-audit, and outline research challenges.

1 Introduction

1.1 Context: the rise of copilots powered by foundation models

In this paper, we use the term *copilot* for any system for human-computer interaction augmented by generative AI. The interaction between human and generative AI may be textual chat, or rely on a graphical interface.

In the past two years, many copilots have become available as commercial tools. In mid 2021, GitHub Copilot was released, aimed at software developers. In late 2022, OpenAI launched ChatGPT, a general-purpose tool for the general public.[1] In early 2023, to list just a few prominent examples, Microsoft released Bing Chat[2] and announced Microsoft 365 Copilot,[3] Google released its Bard chatbot,[4] and OpenAI released its Code Interpreter,[5] one of several plugins for ChatGPT. Meanwhile, researchers have explored copilots, such as Sensecape [1], an interface that provides multilevel abstraction and sensemaking, that illustrate how interaction with a copilot can go beyond textual chat.

Copilots rely on foundation models [2] to generate a response given a prompt provided explicitly or implicitly by the human user. The response may come directly from the model, or indirectly through the use of external tools or plugins. The prompt and response may include more or less structured information such as code in a programming language, lists or tables, or functions to be called, or actions to be executed. In the simplest case of a large language model (LLM), the prompt and response are textual, but in general foundation models may process other modalities such as audio, images, or video.

1.2 The prompt-response-audit cycle

From the perspective of the human user, interaction with the copilot consists of a *prompt-response-audit* cycle, as shown in Figure 1. The cycle is compatible with Glassman's human-AI conversational framework [3].

1. Prompt. Given their implicit *intent* or purpose, the user prepares an explicit *prompt* to pass to the copilot. The user may rely on prompt engineering skills they have learnt, or on prompt engineering tools that help to build the prompt [4]–[6].

PLATEAU
13th Annual Workshop at the Intersection of PL and HCI

Organizers:
Sarah Chasins, Elena Glassman, and Joshua Sunshine

1 *ChatGPT a year on: 3 ways the AI chatbot has completely changed the world in 12 months*, 30 November 2023.
2 *A Tech Race Begins as Microsoft Adds A.I. to Its Search Engine*, 7 February 2023.
3 *Introducing Microsoft 365 Copilot — your copilot for work*, 16 March 2023.
4 *Google CEO tells employees that 80,000 of them helped test Bard A.I.*, 21 March 2023.
5 *ChatGPT plugins*, 23 March 2023.

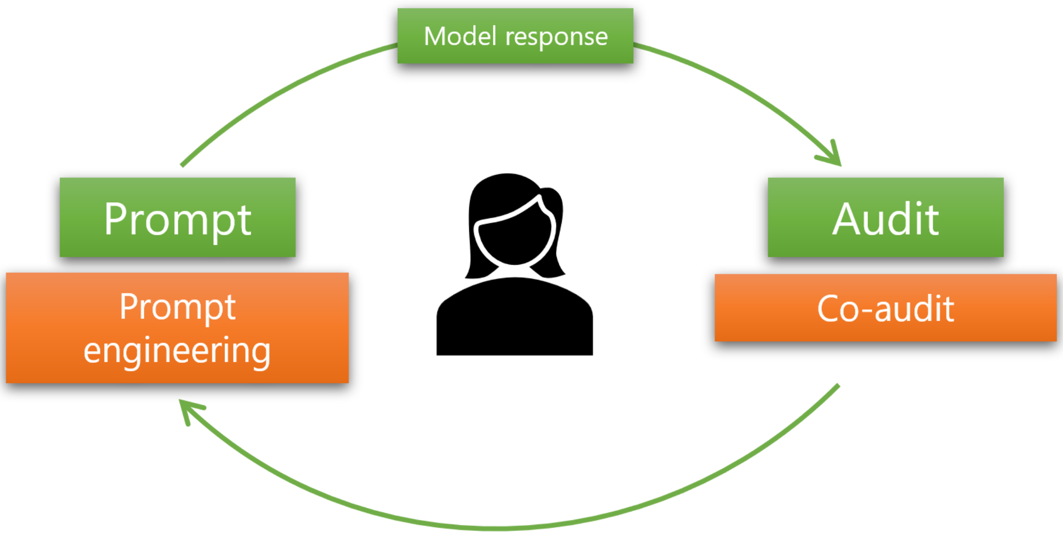


Figure 1. The *prompt-response-audit* cycle.

2. Response. The user waits while, given the prompt, the copilot generates a *model response*. The response may be the direct output from a generative model, or it may be the result of an interaction between a generative model and a set of external tools or plugins. Often, the copilot warns the user that the response may contain errors and should be checked.
3. Audit. Given their intent, the user evaluates the response, and decides to accept it, or to iterate further in the cycle. We call this human evaluation of the response an *audit*, in the sense of a systematic review. Depending on the intent, audit may be to an objective standard, or subjective to the user. During the audit, the user may decide one or more of the following.
   (A) They are satisfied with the response, given the intent.
   (B) There is some mistake in the response, given the intent.
   (C) The intent itself was mistaken, and needs to be refined.
   In case (A), the cycle for this intent ends. In case (B), the user may directly repair the mistake, or do so by continuing with an updated prompt. In case (C), the user continues with an updated intent and prompt, or may directly repair the mistake.

In this context, audit is the systematic review of a piece of AI-generated content, looking for mistakes, that is, mismatches between the intent and the response. The human task of auditing is hard, perhaps often performed poorly and with cursory effort, and getting harder as responses become more complex. Without machine assistance, we fear that audit is often performed poorly by humans. Copilots often warn users that responses may contain mistakes, but checking is a lot to ask of unassisted users.

We introduce the term *co-audit* for any tool-assisted human experience for audit.

Co-audit encourages and assists the user to heed the warnings about AI-generated content, and to do a better job of auditing the response. The "co" in co-audit is to emphasise that the human works with some tool support to do the audit. The tool support may or may not itself use AI technology. Co-audit is about understanding the response, but also about the process of clarifying the intent, and effectively communicating that updated intent to the model.

Figure 1 depicts prompt engineering and co-audit as complementary aspects of human-AI dialogue. The one helps construct the input prompt, while the other helps double-check the output response. We intend the cycle to apply broadly to many different situations, including textual chat between a human and an AI tool, but also mouse-based interactions in graphical interfaces where the prompt is constructed by the system from a mouse click (a form of prompt engineering). Dually, the experience abstracted by the co-audit box in Figure 1 may involve various sorts of interaction with the system, text-based or mouse-based, for example.

For a concrete example, consider ChatProtect[6] [7], a chat experience with a language model, enhanced with features to detect and remove hallucinations. ChatProtect detects hallucination by sampling multiple completions from the model, and testing when they disagree. The co-audit experience lets the user inspect different sentences in the response to see the effect of ChatProtect.

Co-audit tools may help users identify mistakes in their intent, and subsequently refine them (user decision type (C) as above). For example, if the user asks a Copilot to draft a "polite" email, and the model produces a text containing the word "demand"; a co-audit tool might plausibly produce a cautionary warning that this word might not be considered polite. However, on reflection, the user might consider the word acceptable in the context and decide that what they really wanted was a "formal" email. Their intent is thus refined. Observe that the refinement of intent in this case was not a directly designed outcome of the co-audit tool, which is to help assess correctness with respect to the assumed intent of producing a polite email. Instead, it is a beneficial side effect, similar to how the grounded abstraction matching tool studied by Liu, Sarkar, and others [8] was not intended as a tool for *explaining* the model output, but nonetheless could serve as explanations in practice, as was observed during the study. Human intent can be shaped as a side effect of engaging with many tools. Engaging with notations such as algebra or programming languages for instance, provides a notational surface against which user intent is exercised, and within which it can be expressed and negotiated [9]. The same is true of professional knowledge workflows built around the features and limitations of complex software [10]. Even the reification of ideas through the simple act of putting them into words forces us to refine and clarify them. Supporting the refinement of intent, however, is a broad and challenging problem, and there are many ways in which this can be achieved, not all of which involve helping the user check the correctness of the model output. For this reason, we place type (C) decisions aside for the rest of this paper, and while noting the potential of co-audit tools to help refine user intent, we will not discuss in further detail how to explicitly support this function. We instead focus on the relatively simpler, but nevertheless challenging and important case where the user intent is "correct" but the response mistaken.

1.3 The purpose of co-audit is to identify mistakes and repair them

When auditing, the user is looking for any *mistake*: any part of the response that doesn't match their intent. Mistakes include textual hallucinations, defined as "generated content that is nonsensical or unfaithful to the provided source content" [11]. Other examples (which may or may not count as hallucinations) include a summary that omits important information, faulty claims about the world, a faulty fictional narrative that mixes up characters or timelines, faulty mathematical reasoning (including faulty arithmetic or algebra), faulty code, or a faulty image (such as a hand with six fingers).

Our definition of mistake includes correct information in a response that doesn't match the intent; for example,"Cairo is the capital of Egypt" would be a mistake given the prompt "What is the capital of Eritrea". Mistakes may be useful even if they don't match the intent behind the prompt. In our example, the user may be compiling a list of capital cities in Africa.

There are several concrete examples where copilots make mistakes of this sort. For instance, GPT-4 can make mistakes in logical and arithmetic reasoning [12]. OpenAI's Code Interpreter is effective at automating data science tasks [13]. Still, users need to be careful: a commentator noted that "Code Interpreter made several logical mistakes that only an expert could have caught."[7] In the domain of mathematics, Collins and others develop a framework to evaluate how effectively humans can develop proofs despite the faulty algebraic reasoning made by language models [14]. One of their conclusions for any mathematician using an LLM is to "pay attention": LLMs can generate extremely compelling mathematical language and still be wrong.

Bias, defined generally as a disposition or prejudice towards or against a person or group, particularly when such a disposition impacts the perception of fairness, may also be a form of mistake. Bias in an LLM's output may arise from bias in the training data, the training and inference algorithms themselves, the user prompt, or a combination. The questions of blame (where does bias come from)

6 *ChatProtect detects and removes hallucinations of LLMs when you chat with them*, 9 September 2023.
7 *I rewrote around 1,000+ using OpenAI's Code Interpreter and am impressed!*, 12 July 2023.

and responsibility (what, if anything, to do about it, and who should do it) are heavily debated.

What position should co-audit tools occupy with respect to bias? To answer this question, it is helpful to consider a concrete example. Suppose the user has a collection of documents pertaining to a complex topic involving dispositions towards different groups, such as affirmative action at universities. The collection may include essays, news articles, university memoranda, legal proceedings, etc. Suppose the user wishes to summarise these documents and prompts an LLM to "Give me a summary of these documents". In response, the LLM produces a summary which can be interpreted as prejudiced against a particular group. Is this a mistake? Consider some possible nuances of this situation:

- The word "summary" as applied by the user may implicitly include the property of being free from bias against a particular group, on the basis that a good summary should be unbiased and thus no further elaboration is needed in the prompt (i.e., the user sees no need to explicitly specify "give me an *unbiased* summary"). In this case, a biased summary is a mistake. On the other hand, the word "summary" may imply no such requirement – here the biased summary may not be considered by the user to be a mistake. Should co-audit tools intervene here to introduce the system designers' values around biases?
- The collection of documents might itself be biased towards the views of one particular person or group, and the bias in the output is a result of this. The output could be interpreted as an accurate summary of this particular biased set of documents. Is this bias a mistake? It might not be, if the user wished for a summary that accurately reflected the biases in their corpus.
- The language model is itself biased towards the views of a particular group because of its training data or algorithm, and the bias in the output is a result of this. This would clearly seem to be an undesirable error. However, it might still not count as a mistake if it matches the user intent. What if the user was explicitly aware of the bias in the model and wished to apply it to produce biased content? Again, should co-audit tools intervene here to introduce the system designers' values around biases?

Clearly, whether bias is a mistake depends on the user intent, which we can now see is woefully underspecified in our initial vignette of the situation. A critical design approach might venture that we ought to shape the user's intent in such a way that their intent includes the property of being free from bias, regardless of the source of the bias. However, we have already left intent shaping outside the explicit scope of co-audit. We treat bias similarly: if freedom from bias is part of the user intent, and the output is biased, then this is a mistake and within the remit of co-audit. Co-audit tools are involved in rectifying bias to the extent that we help the user match the output to their intent, but no further.

The user faces three challenges when auditing the response. These are intensifying as more knowledge workflows increasingly incorporate AI-generated content.

1. *It is hard to entirely eliminate mistakes at the model response step.*
   Despite much research [15]–[17] and benchmarks [18] hallucinations still occur. Copilots built on natural language generation (and indeed other modalities such as images) therefore inherit the possibility of hallucinations and other kinds of error.
2. *The patterns of mistakes from humans and AI are different.*
   The shift from human-generated to AI-generated content creates a qualitative shift in the user experience of audit. For example, checking a human-authored document summary for correctness is done with a model of the author's intent and kinds of quality issues that are known to arise when people write text. Is the author well-meaning or adversarial? Are we looking for unsupported statements, irrelevant statements, omissions, or overemphasis? In contrast, AI-generated content has errors for which it is impossible to ascribe a "motive". Due to hallucinations, there are broad issues of factual correctness and grounding in the source data that are more common and important for AI-generated content than for human-generated content.
3. *The user needs decision support after finding a mistake in the response.*
   A mistaken response might be viewed qualitatively as an error in the system's interpretation or understanding of the intent, or as an error in executing the interpretation. Depending on how the mistake is interpreted, the user may need to update their prompt, attempt to work with and refine

the mistaken output, or abandon the generative tool altogether for a particular task.

Intent matching is the user challenge of expressing their intent in a way that the model would "understand" correctly, and then checking whether it has done so (generalising "abstraction matching" [8]). When knowledge work is dominated by primarily human-generated content, intent matching is externalised to the relationship between individuals; if I ask you to do a task, and it appears that you had misunderstood me, I will take it up with you, not the software. When an individual works alone, the issue of intent matching is externalised to the user's own conception of their skills ("am I applying this feature correctly to achieve what I want?"). But generative AI tools create a new expectation, that the user can express their intent in the way they might express it to a colleague. This creates a new type of relationship between user and system in which the responsibility for intent matching cannot fall either entirely on the user, or be externalised to the relationship with a human colleague.

The research field of interactive machine learning commonly frames the role of the user interface as a decision support mechanism; i.e., giving the user the information they need to take the next step in refining and improving their model [19] [20]. Co-audit systems need to incorporate decision support specifically geared towards repairing issues with intent matching and correctness, a type of support that is only necessary in a knowledge workflow dominated by AI-generated content.

1.4 Co-audit matters in some application areas more than others

Co-audit experiences matter most in application areas where correctness is important and factual or reasoning errors are consequential.

Three such example areas are technology, healthcare, and finance. Table 1 shows examples of tasks within each that could benefit from co-audit experiences. The table mentions three examples of AI systems that could potentially benefit from co-audit: GitHub Copilot, Nuance[8] [9], and systems from Bloomberg. The tasks are not necessarily applicable to all the systems mentioned.

The need for co-auditing for exploratory or creative areas, such as creative writing or music recommendations, is less clear as there is no definite right answer, and it is subjective whether part of the response is a mistake. Still, a critique that identifies shortcomings is valuable as a co-audit experience. If an LLM helps to write a novel, a co-audit experience could help identify shortcomings in the story or writing, and to identify potential repairs. The role of tool support in such a scenario, rather than co-audit to detect mistakes, could be to provide the user an estimate of the system's diversity and encourage the user to reassess their asks or priorities.

Hence, the need for co-audit varies depending on properties of the users, the domain, and the task, making co-auditing a challenging and highly custom research space. There is not likely to be a single co-audit experience that dominates all application areas. Yet it seems plausible that certain principles of design for co-audit experiences may be broadly applicable across different domains.

1.5 Co-audit matters to some people more than others

As we design and critique co-audit mechanisms, we should bear in mind that using generative AI demands deep cognitive effort. Tankelevitch and others [21] consider thinking, evaluating, and adapting as the crucial aspects to the user's cognitive effort as they work with generative AI. Their work is informed by the psychological concept of metacognition (thinking about thinking) [21].

1. Thinking: The user needs self-awareness of their concrete goals and intentions, and the ability to express their goals as effective prompts or commands.
2. Evaluating: The user needs the ability to evaluate the output, and the confidence to challenge it when it appears wrong.
3. Adapting: The user needs flexibility either to repeat and to adapt prompts, or to correct the output.

Thinking is the effort during the prompt part of the prompt-response-cycle in Figure 1, whereas evaluating and adapting are the effort when using a co-audit tool. Evaluating outputs from generative

8 *How can AI unlock next-generation radiology reporting?*, 15 August 2022.

9 *Nuance and Microsoft Announce the First Fully AI-Automated Clinical Documentation Application for Healthcare*, 20 March 2023.

AI is far more important and effortful compared to, for example, mere word or phrase suggestions from auto-complete powered by previous generations of machine learning technology. The user's confidence in themselves is also important: studies of decision-making with AI find that users' confidence in themselves may be more important than their confidence in the AI itself [22]–[24].

Hence, co-audit mechanisms may matter most to less intrinsically confident users: co-audit can equip them to effectively and efficiently assess and incorporate suggestions from generative AI.

1.6 This paper and its context

The contributions of this paper are to identify co-audit experiences as an important aspect of interactions between humans and copilots, and to begin a systematic study of co-audit, including needs, case studies, some guiding principles, risks, and research questions. Co-audit is a novel approach to improve the usability and reliability of code generated from natural language by LLMs.

This paper came about as a result of a one-day internal workshop at Microsoft in May 2023. A companion technical report includes further details and examples [25]. The work in this paper contributes towards the Microsoft New Future of Work Report 2023 [26].

Area	Example AI Systems	Example Tasks
Technology	GitHub Copilot	project planning [27]
		code authoring [28]
		documentation [29]
		code reviews [30]
		analysis [31]
Healthcare	Nuance	diagnosis [32]
		prescription calculations
		authoring clinical notes [33]
		nutrition recommendations [34]
		mental health [35]
		on the go recommendations
Finance	Bloomberg GPT	financial research [36]
	[37]	trend analysis [38]
	FinGPT	insurance policy recommendations
	[39]	financial investments [40]
		business plans [41]

Table 1. Important areas for co-audit. Specialised models and copilots have started to emerge for technology, healthcare and financial applications. There is a need and opportunity for co-audit tools in key tasks, such as the ones presented in this table.

2 Taxonomy of co-audit needs

The design requirements for co-audit systems likely vary substantially across different task contexts. To make progress in accounting for the diversity of human-AI collaboration scenarios, we propose a preliminary taxonomy that characterizes key dimensions along which co-audit needs differ.

Ease of detection: how easy is it for the user to detect an error in the AI output? This can range from very easy (obvious at a glance) to hard (e.g., needs careful inspection of the output, cross-referencing with other sources, etc.). For instance, correctness of an AI-generated image usually can be told at a glance. On the other hand, correctness of several hundred lines of Python performing data analysis may be far from obvious.

Cost of error: how costly is an error in this situation? The cost can range from low (the error is inconsequential) to high (catastrophic). Automatically assessing the cost is a difficult task for tools as it depends on rich context. For example, AI introducing a formula error can be relatively inconsequential if the formula is being used as part of an exploratory analysis with no downstream uses, but the same kind of error can be catastrophic if its result is used to make a critical business decision.

Frequency of audit: how often is co-audit required or needed in the workflow? If the aim is to provide spot checking of every transient AI-generated output, the design of co-audit tools will need to be lightweight and impose low cognitive and information processing burdens, focusing only on key aspects of intelligibility (akin to spelling and grammar checking). If errors are more infrequent, or if co-audit only occurs at significant milestones (e.g., at the end of the preparation of a document, or for formal auditing procedures) then co-audit tools can be more comprehensive and information dense.

Solution definition and constraints: how large and well-defined is the solution space? A solution space may be very large, yet still well-defined: for example, if an LLM is asked to generate a sorting algorithm, there are a potentially infinite number of correct solutions, but it can be established definitively whether a given solution is correct. In some cases the solution may take a very wide range of acceptable forms, but which are judged subjectively (e.g., creating a nice looking design for a presentation, or writing an essay). Conversely, solutions may have very narrow or even singular objectively correct form (e.g., the items in a budget are accounted for correctly in a grand total).

End-user expertise: what kinds of skills and expertise does the intended end-user have? For instance, co-audit for a code-generating copilot would vary depending on whether the end-user is a professional programmer, versus a non-expert end user. Analogous situations arise in other application domains: in healthcare, for instance, a system for supporting treatment and diagnosis would present different information (and different audit challenges) for doctors seeking to make a decision grounded in clinical practice, versus patients with limited medical expertise.

Positioning: What is the implied metaphorical relationship between the AI copilot and the human user? For example, the pair programming framing in GitHub Copilot implies co-working towards a shared goal, with tasks delegated from the user to Copilot. In contrast, meeting coach in Microsoft Teams or presenter coach in Microsoft PowerPoint position themselves as reflective critics of the users' behavior. Different AI "personas" require different kinds of audit support. For a co-working AI like GitHub Copilot, the audit process would likely need to be focused on ensuring that the AI is enhancing the performance and skillset of the user—providing efficient, correct and relevant suggestions. For an AI with a critical role such as a coach in Teams or PowerPoint, the audit mechanism might focus more on evaluating its ability to provide constructive and accurate feedback.

Scale: how can we design co-audit to scale to large amounts of AI-generated content? Leveraging AI to generate content empowers individuals to be more productive. As a result, AI will likely allow more content to be generated more quickly, resulting in the challenge of having individuals co-audit all that content. Co-audit tools must be designed to scale in the amount of human effort as the amount of generated content increases.

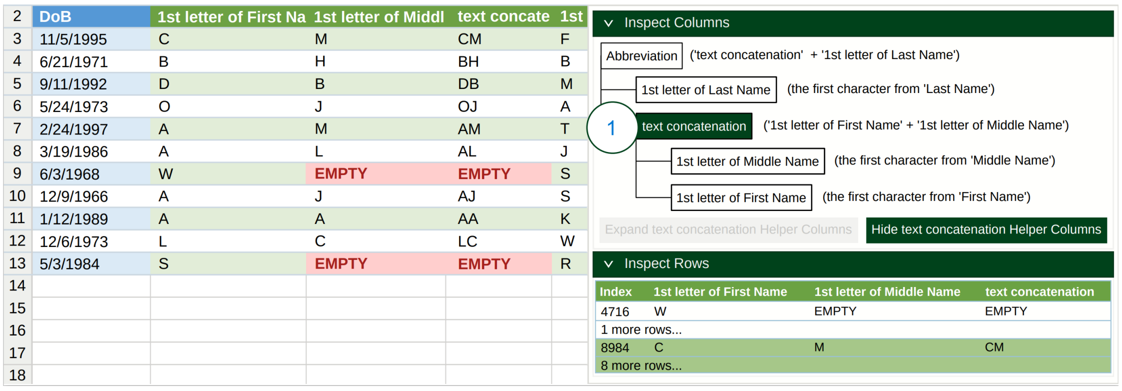
Collaboration: how do we design co-audit to support collaboration? Section 1.2 frames audit as an activity by a single human using an AI tool, and co-audit as tool assistance for audit. More generally, two or more humans could collaborate during the audit step. A co-audit tool could enlist the help of a co-worker to help inspect or repair the response from the AI. The co-worker could be a local colleague or a remote gig worker. For instance, if applied to consider AI-generated content, the Find-Fix-Verify crowd programming pattern [42], which enlists gig workers to help within a productivity tool, would be an example of co-audit.

3 Case study: co-audit tools for spreadsheet computations

We describe two related tools [8] [43] built at Microsoft Research as part of an internal project to synthesise spreadsheet code from natural language. The tools shared infrastructure to generate Python using the OpenAI Codex model and to execute it using an addin for Microsoft Excel. Both these tools have co-audit experiences intended to help the end user double-check the generated code.

3.1 Grounded abstraction matching

Grounded abstraction matching is a technique, implemented in one of our tools [8], to help users learn how to effectively communicate their intent to AI systems like code-generating large language models. It addresses the abstraction matching problem, where users struggle to find the right level of


Inspect Columns
1
Expand text concatenation Helper Columns
Hide text concatenation Helper Columns
Inspect Rows

	DoB	1st letter of First Na	1st letter of Middl	text concate	1st
3	11/5/1995	C	M	CM	F
4	6/21/1971	B	H	BH	B
5	9/11/1992	D	B	DB	M
6	5/24/1973	O	J	OJ	A
7	2/24/1997	A	M	AM	T
8	3/19/1986	A	L	AL	J
9	6/3/1968	W	EMPTY	EMPTY	S
10	12/9/1966	A	J	AJ	S
11	1/12/1989	A	A	AA	K
12	12/6/1973	L	C	LC	W
13	5/3/1984	S	EMPTY	EMPTY	R
14					
15					
16					
17					
18					

Abbreviation ('text concatenation' + '1st letter of Last Name')
1st letter of Last Name (the first character from 'Last Name')
text concatenation ('1st letter of First Name' + '1st letter of Middle Name')
1st letter of Middle Name (the first character from 'Middle Name')
1st letter of First Name (the first character from 'First Name')

Index	1st letter of First Name	1st letter of Middle Name	text concatenation
4716	W	EMPTY	EMPTY
1 more rows...			
8984	C	M	CM
8 more rows...			

Figure 2. ColDeco

abstraction when expressing their goals in natural language that the LLM can reliably interpret.

Grounded abstraction matching works by taking the user's initial natural language query, generating code from it via the LLM, and then mapping that code back into a natural language utterance that serves as an editable example of how to invoke the same behavior from the system. Concretely, the user enters a query like "calculate average monthly sales." The system generates Python code from this query, e.g., `df[`monthly_sales_avg'] = df[`sales'] / 12`. Then it translates this code back into a grounded utterance:

1. Create column `monthly_sales_avg`
2. Column `sales` divided by 12

This grounded utterance represents the level of abstraction—vocabulary, structure, specificity—needed for the system to reliably reproduce the same output. The user can edit the steps and observe the effects on the code produced, developing a mental model of how to effectively prompt the system.

Grounded abstraction matching exemplifies co-audit by assisting users in evaluating AI-generated content in multiple ways. For example, the grounded utterance reveals how the system interpreted the original query, making mismatches in intent clear. Users can update the utterance to align intent. The step-by-step breakdown of the Python code into naturalistic utterances exposes the reasoning chain, helping non-expert users spot faulty logic. Finally, grounded utterances provide an editable target to iteratively refine queries, which gives users the decision support then need to proceed after they have received the model output.

In summary, grounded abstraction matching helps users learn how to effectively communicate with LLMs for code generation. By grounding output code back into editable, step-by-step natural language utterances, it provides a mechanism for users to co-audit the system's interpretation of their intent, spot errors, and iteratively repair issues, while exposing them to reliable patterns of interaction.

3.2 An end-user spreadsheet inspection tool for AI-generated code

ColDeco [43] is a spreadsheet tool for inspecting and verifying calculated calculated columns without requiring the user to view the underlying code. With generative AI, natural language is now a programming language, meaning users without coding experience can generate complex logic. ColDeco uses *helper columns* and *row summaries* to assist users in verifying that the program behaves correctly, or to localize the fault if it does not.

Figure 2 presents ColDeco as a user debugs a generated program intended to calculate an abbreviation composed of the initial letter from the first, middle, and last names.

The *inspect columns* view displays the generated solution which has been decomposed into helper columns by the user; each generated column has a natural language description of the code associated with the column. The columns are arranged in a tree view representing the underlying program structure, with Label 1 annotating the intermediate column *text concatenation*. The helper column combines the initial letters from the first and middle names, and by expanding the column in ColDeco, the new columns are added to the table.

The *inspect rows* view clusters rows in the table according to evaluation behaviours that are determined through dataflow analysis. There are two clusters for this table: the rows that return a string for the abbreviation, and the rows that erroneously calculate a missing value.

Like grounded abstraction matching, ColDeco satisfies co-audit needs in multiple ways. For example, helper columns and their natural language descriptions illustrate how the system has implemented the solution, allowing users to verify that it matches their intent. Row summaries allow users to quickly identify rows that may need further inspection. Finally, expanding helper columns can enable users to make targeted repairs through *programming-by-example* techniques such as FlashFill [44].

4 Principles of UX design for co-audit

Here is a list of design principles distilled from discussions at our workshop and afterward. The participants were all Microsoft employees. Some were from product teams engaged in building and evaluating copilots. Some were from research teams experienced with research using LLMs. The first and second authors drafted these principles, attempting to reflect the most salient points from the discussion. All participants were invited to help edit and refine these principles, and be co-authors, although not all did.

1. *Don't rely just on the LLM to co-audit itself.*
   It's an anti-pattern to trust the language model to audit itself. For example, ChatGPT created a legal motion for a lawyer, but with made-up cases, rulings and quotes. In an attempt at co-audit, the lawyer asked the program to verify that the cases were real. ChatGPT responded incorrectly that the "cases I provided are real and can be found in reputable legal databases". The lawyer submitted the brief to court and consequently was fined by the judge.[10] The example illustrates that co-audit may fail: an attempt at co-audit may or may not succeed in detecting an error.
2. *Ground outputs by citing reliable sources.*
   Citing sources is becoming a common pattern for AI-based chat systems[11] [12] and their plug-ins, such as the Wikipedia plug-in for ChatGPT.[13]
3. *Teach the user how to construct effective prompts.*
   For example, apply grounded abstraction matching: after translating the user query into a system action, help the user understand how to consistently invoke the same system action through an editable example, e.g., a "grounded utterance" [8].
4. *Inform the user with visuals as well as text.*
   For example, ConceptEVA provides co-audit of a document summary using techniques from exploratory visual analysis to show the concepts of interest underpinning the summary [45].
5. *Get the user to choose between multiple options.*
   Consider, for example, the LEAP environment for Python programming [46]. LEAP uses comments or code context to suggest multiple AI-generated suggestions. Its co-audit features include preview of each suggestion, and live editing of the suggestion including after its insertion into the program.
6. *Prioritize the user's attention to most likely error.*
   For example, consider an experimental LLM-powered search experience [47] that uses colors to highlight high or low confidence measurements (numbers with units) as a co-audit feature for search results. The highlighting relies on token probabilities from the LLM. The authors find overall that highlighting reduces the time taken for users to spot incorrect information.
7. *Guide the user to attend to links between parts of the output and the prompt or input document.*
   Take, for example, the nl2spec system [48]. Given a natural description of a logical property, the system uses an LLM to synthesise a formula for the property in linear temporal logic. To aid co-audit, the system additionally uses the LLM to map subformulas of the output formula back to corresponding natural language fragments of the input. Users can edit the subtranslations to correct errors.
8. *Involve two or more humans in a collaborative experience.*
   For example, one user may instigate the AI-generation of an output, while a different user may

10 *Here's What Happens When Your Lawyer Uses ChatGPT*, 27 May 2023.
11 *Reinventing search with a new AI-powered Microsoft Bing and Edge*, 7 February 2023.
12 *Bard can now connect to your Google apps and services*, 19 September 2023.
13 *New Wikipedia ChatGPT plugin*, 13 July 2023.

help to co-audit the output. Here, we go beyond the prompt-response-audit cycle in Figure 1, which revolves around a single user, to be a collaboration amongst two or more humans, assisted by co-audit tools.

9. *Exploit tools designed for checking human-generated content.*
   In Section 3 we described a couple of research prototypes designed to co-audit AI-generated spreadsheet computations. Still, if the computation is represented as a formula, any existing debugger for a spreadsheet formula, such as FxD [49], can serve as a co-audit tool.
10. *Aim for positive "weekly cost-benefit" ratio for user time invested.*
    This principle concerns measuring adoption of co-audit technology and using the measurement as a guide to ensure that co-audit is worth the time invested by users. (This final principle, more about adoption than design, did not originate at our workshop, but is adapted from a proposal for successful adoption of formal methods technology [50].)

5 Potential pitfalls for co-audit: some research challenges

In this section we consider potential risks arising from co-audit tools that can lead to attacks, system misuse (under-reliance or over-reliance) and even long-term societal harms. The content is derived from an internal workshop and represents the views of the participants, as described at the start of Section 4. This section is not meant to provide an exhaustive overview, but we believe that starting to address the outlined issues is important for the co-audit experience. These risks are well known in the literature for related domains. We encourage the community to help us validate our hypothesis that they emerge for co-audit mechanisms as well, assess whether our proposed mitigations are valid and identify further gaps in our thinking.

Security Attacks: The use of co-audit may help mitigate potential security attacks (and other responsible AI concerns, such as privacy and fairness issues) by facilitating human oversight of the generated content for possible harms. At the same time, the co-audit experience itself can be another possible point of attack for adversaries. For example, attackers understanding the abilities of co-audit can formulate their attacks specifically to target co-audit weaknesses. Also, the co-audit tool itself, if compromised, can be a rich target for exploitation as it has direct access to user information, the AI-generated content, etc. Further research is needed to identify attacks and mitigations.

System under-reliance or over-reliance: We believe that co-auditing tools can be susceptible to both under-reliance and over-reliance. While co-audit gives the user a greater understanding of the AI-generated content, it does not guarantee that the result is correct. Likewise, if the cognitive load of co-auditing is too high, users could under-rely on the co-auditing system (for example, by having too many tasks to verify) or if they lack trust in the system. To minimise the cognitive load, the system should be non-intrusively integrated in the user's flow and ask for very specific input from the human to make corrections. To improve trust, the system could show the work or rationale at various granularities including the impact of fixes (if we make a change at a certain point in the flow, be explicit about how the remainder of the flow gets impacted) and be explicit about the coverage of types of errors it can handle. Users could also over-rely on the co-auditing system if the system exhibits anthropomorphic behaviour (for example, by generating content claiming personified attributes) or if the remit and level of accuracy of the system are not clearly understood. Using linguistic cues to show uncertainty and articulating the specific assumptions made during the audit process could help the user understand the system's limitations.

Societal harm: There are significant long-term implications as humans increasingly leverage AI in their activities. Co-audit, which provides users with greater confidence in their AI interactions, may help accelerate this trend. How human/AI interaction will evolve and what implications this evolution has for society are largely unknown. This uncertainty requires careful consideration of potential risks and unintended consequences that more powerful AI and better co-audit tools may enable. For example, we could end up reducing creativity or diversity of thought by incentivising humans to produce outputs that co-audit systems can verify. There may also be an impact on workers (e.g., by turning every professional into their lesser-paid gig-based equivalent as users of co-audit tools), resulting in deskilling of workers, reduction of their economic value, and loss of jobs. In this scenario, there is a potential harm in the transfer of responsibility of review to the LLM with co-audit tool as the LLM is given more and more responsibility as its capabilities increase. There is also the risk of a vicious cycle of skill-transfer from the human to the LLM as the LLM directly learns from successful human interactions and requires less and less guidance from human input over time. Moreover, the availability of AI and co-audit may result in humans losing their ability to effectively reason about knowledge sources and make judgements about goodness and correctness. Use of AI and co-audit may lead to confusion around who is responsible for an error that leads to harm. For co-auditing tools we believe that we need to have in place opt-in/opt-out functionality and sufficient customization such that the user can control the level of correctness, verbosity, and modality.

6 Related work

6.1 Related but distinct concepts

Algorithmic audit is defined by Metaxa and others [51] as "a method of repeatedly querying an algorithm and observing its output in order to draw conclusions about the algorithm's opaque inner workings and possible external impact". Rastogi and others [52] argue for the importance of rigorous algorithmic audit of LLMs before deployment, to counter bias and resulting harms. Co-audit is distinct from algorithmic audit: co-audit examines a single response rather than repeatedly prompting the algorithm to examine its statistical properties. Still, there may be scope to detect bias in responses via co-audit mechanisms.

Co-audit is also distinct from *evaluation* of an NLP system's performance as a whole, for instance, on a suite of benchmarks. See the classic review by Spärck Jones and Gallier [53], for example. The distinction is that co-audit is for a specific instance of a user interaction, while algorithmic audit or NLP evaluation consider general properties of the whole system or model, rather than individual actions.

An alternative definition of co-audit might be that co-audit is a human and AI auditing their work together. This metaphor presents human and AI as two entities who, while collaborating to produce an output, audit the output together. Although anthropomorphic metaphors are common, the practice may be misleading [54]. Hence, we prefer to frame co-audit in terms of tool use by a human user, instead of a metaphorical collaboration [55] [56].

6.2 Mistakes

Dealing with errors has been a key part of research on human-AI interaction; for instance, several of the guidelines for human-AI interaction [57] concern how to deal with AI errors. [58] identify research challenges raised by LLMs for AI transparency, citing research explaining individual outputs [59]–[63].

The term "explanation" encompasses a wide variety of techniques to assist the user of AI systems, and subsumes aspects of the experience that relate to the correctness of model outputs, such as evaluating model outputs and then correcting them. Research in explainable AI has thus long encompassed the specific subset of concerns we delineate here as the need for co-audit. For example, Lim and Dey's intelligibility types for context-aware applications [64] include types of information that help users detect errors (e.g., why, how, why not, what if, certainty), as well as correct them (control). In Tintarev and Masthoff's criteria for explanations [65], co-audit is covered by *transparency* (explain how the system works) and *scrutability* (allow users to tell the system it is wrong).

While the concerns of co-audit are not new, the broad introduction of generative AI with its particular strengths and limitations is accompanied by a number of shifts in the nature of the user experience of intelligent systems. The shift in user experience has been described as the "generative shift" [66]. In particular, users will apply AI more intensively to existing tasks, they will apply it more extensively to tasks which they previously did not apply AI to, and they will apply it more frequently.

There are concomitant shifts in the user experience of *error*. System errors will have the opportunity to become more consequential, span across a wider set of tasks, and occur more frequently than before. In earlier generations of interactive machine learning (IML) tools, the user was responsible for directly training the model to shape its behaviour and make it perform more optimally in the long run, e.g., the user of a photo editing tool training an image segmentation engine to remove the background from their images [67], a spreadsheet user building a model of their data to impute missing values [68], [69], or an email client user training a spam filter [70]. In contrast, the prevailing commercial idioms of user experience for contemporary generative AI tools are far more passive: users can control the behaviour of the model in individual instances through their prompts, but they cannot participate in its construction on the more fundamental level that was common in earlier IML tools, such as through the curation of the training set, provision of reinforcement learning feedback signals, or adjustment of training hyperparameters. Consequently, the interaction context inhabited by Lim and Dey's "control" [64], and Tintarev and Masthoff's "scrutability" [65], is increasingly divergent from the interaction context inhabited by generative AI tools: we thus offer *co-audit for copilots* as a practical reinterpretation of those concepts.

7 Conclusion and call to action

Just as individuals learned how to best interact with search engines over a period of years, they will have to learn how best to interact with generative AI and copilots. Human skills in both prompt engineering and audit are needed. We have discussed a range of aspects of co-audit tools to help and teach people to audit AI outputs. We believe that some of these co-audit tools and skills should exist across copilot experiences while other skills will be domain-specific. Our goal is to jump-start research into these co-audit experiences to maximize the benefits of using AI while minimizing risks. As a starting point, we have outlined principles of UX design for co-audit and provided examples to illustrate how these principles can be applied in practice. But many questions and important research challenges remain. Co-audit experiences will evolve as the underlying foundation models and copilot infrastructures will evolve. But because co-audit directly connects with user, it is especially important that these experiences remain similar even as the technology changes. To ensure that we create enduring co-audit experiences, we challenge the research community to address these questions:

- How to integrate a co-audit user experience into copilot chats?
- How to integrate the co-audit experience across applications that relate to the same content?
- How can co-audit help individuals create, maintain, and check increasing volumes of content that AI generation enables?
- What co-audit skills are transferable across different vertical uses of AI?
- Will the principles of understanding the co-audit experience remain stable as AI evolves or will it require frequent redesign?

Acknowledgements

Thanks to Ashley Llorens, Brittney Muller, and all at Microsoft Research Outreach for supporting the workshop reported in this paper. Gavin Abercrombie provided technical assistance. Christian Canton offered useful advice and suggestions. We are grateful to Peter Lee for his comments and support, and to Kevin Scott for announcing our research on co-audit in the Verge.[14]. We thank the participants and our mentors at the PLATEAU workshop in February 2024 for their most useful and thoughtful feedback. And finally, we acknowledge Bill Gates for his encouragement of this work. Thank you for the opportunity.

References

[1] S. Suh, B. Min, S. Palani, and H. Xia, *Sensecape: Enabling multilevel exploration and sensemaking with large language models*, 2023. arXiv: 2305.11483 [cs.HC].

[2] R. Bommasani, D. A. Hudson, E. Adeli, *et al.*, *On the opportunities and risks of foundation models*, 2022. arXiv: 2108.07258 [cs.LG].

[3] E. L. Glassman, *Designing interfaces for human-computer communication: An on-going collection of considerations*, 2023. arXiv: 2309.02257 [cs.HC].

[4] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, *Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing*, 2021. arXiv: 2107.13586 [cs.CL].

14 *Microsoft CTO Kevin Scott thinks Sydney might make a comeback*, 23 May 2023.

[5] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why Johnny can't prompt: How non-AI experts try (and fail) to design LLM prompts", *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3544548.3581388.

[6] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers, *In-IDE generation-based information support with a large language model*, 2023. arXiv: 2307.08177 [cs.SE].

[7] N. Mündler, J. He, S. Jenko, and M. Vechev, *Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation*, 2023. arXiv: 2305.15852 [cs.CL].

[8] M. X. Liu, A. Sarkar, C. Negreanu, *et al.*, "'What it wants me to say': Bridging the abstraction gap between end-user programmers and code-generating large language models", in *CHI*, ACM, 2023, 598:1–598:31. [Online]. Available: https://www.microsoft.com/en-us/research/publication/what-it-wants-me-to-say-bridging-the-abstraction-gap-between-end-user-programmers-and-code-generating-large-language-models/.

[9] K. E. Iverson, "Notation as a tool of thought", in *ACM Turing award lectures*, 2007, p. 1979.

[10] A. Sarkar, "Should computers be easy to use? questioning the doctrine of simplicity in user interface design", in *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '23, Hamburg, Germany: Association for Computing Machinery, 2023, ISBN: 9781450394222. DOI: 10.1145/3544549.3582741. [Online]. Available: https://doi.org/10.1145/3544549.3582741.

[11] Z. Ji, N. Lee, R. Frieske, *et al.*, "Survey of hallucination in natural language generation", *ACM Comput. Surv.*, vol. 55, no. 12, 248:1–248:38, 2023. [Online]. Available: https://doi.org/10.1145%2F3571730.

[12] S. Bubeck, V. Chandrasekaran, R. Eldan, *et al.*, *Sparks of artificial general intelligence: Early experiments with GPT-4*, 2023. arXiv: 2303.12712 [cs.CL].

[13] L. Cheng, X. Li, and L. Bing, *Is GPT-4 a good data analyst?*, 2023. arXiv: 2305.15038 [cs.CL].

[14] K. M. Collins, A. Q. Jiang, S. Frieder, *et al.*, *Evaluating language models for mathematics through interactions*, 2023. arXiv: 2306.01694 [cs.LG].

[15] N. Lee, W. Ping, P. Xu, *et al.*, "Factuality enhanced language models for open-ended text generation", in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 34 586–34 599. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/df438caa36714f69277daa92d608dd63-Paper-Conference.pdf.

[16] K. Li, O. Patel, F. Viégas, H. Pfister, and M. Wattenberg, *Inference-time intervention: Eliciting truthful answers from a language model*, 2023. arXiv: 2306.03341 [cs.LG].

[17] Y.-S. Chuang, Y. Xie, H. Luo, Y. Kim, J. Glass, and P. He, *Dola: Decoding by contrasting layers improves factuality in large language models*, 2023. arXiv: 2309.03883 [cs.CL].

[18] T. Gao, H. Yen, J. Yu, and D. Chen, *Enabling large language models to generate text with citations*, 2023. arXiv: 2305.14627 [cs.CL].

[19] I. Nunes and D. Jannach, "A systematic review and taxonomy of explanations in decision support and recommender systems", *User Modeling and User-Adapted Interaction*, vol. 27, pp. 393–444, 2017. [Online]. Available: https://arxiv.org/abs/2006.08672.

[20] A. Sarkar, "Is explainable AI a race against model complexity?", in *Workshop on Transparency and Explanations in Smart Systems (TeXSS), in conjunction with ACM Intelligent User Interfaces (IUI 2022)*, ser. CEUR Workshop Proceedings, Mar. 2022, pp. 192–199. [Online]. Available: http://ceur-ws.org/Vol-3124/paper22.pdf.

[21] L. Tankelevitch, V. Kewenig, A. Simkute, *et al.*, *The metacognitive demands and opportunities of generative AI*, 2023. arXiv: 2312.10893 [cs.HC].

[22] L. Chong, G. Zhang, K. Goucher-Lambert, K. Kotovsky, and J. Cagan, "Human confidence in artificial intelligence and in themselves: The evolution and impact of confidence on adoption of AI advice", *Computers in Human Behavior*, vol. 127, Feb. 2022. [Online]. Available: https://doi.org/10.1016/j.chb.2021.107018.

[23] M. Steyvers and A. Kumar, "Three challenges for AI-assisted decision-making", *Perspectives on Psychological Science*, 2023. [Online]. Available: https://doi.org/10.1177/17456916231181102.

[24] H. Tejeda, A. Kumar, P. Smyth, and M. Steyvers, “AI-assisted decision-making: A cognitive modeling approach to infer latent reliance strategies”, *Computational Brain & Behavior*, vol. 5, pp. 491–508, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s42113-022-00157-y.

[25] A. D. Gordon, C. Negreanu, J. Cambronero, *et al.*, “Co-audit: Tools to help humans double-check AI-generated content”, Microsoft Research, Oct. 2023. [Online]. Available: https://aka.ms/co-audit.

[26] J. Butler, S. Jaffe, N. Baym, *et al.*, “Microsoft new future of work report 2023”, Microsoft, Tech. Rep. MSR-TR-2023-34, Dec. 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/microsoft-new-future-of-work-report-2023/.

[27] M. Schroder, *AutoScrum: Automating project planning using large language models*, 2023. arXiv: 2306.03197 [cs.AI].

[28] B. Rozière, J. Gehring, F. Gloeckle, *et al.*, *Code Llama: Open foundation models for code*, 2023. arXiv: 2308.12950 [cs.CL].

[29] J. Y. Khan and G. Uddin, “Automatic code documentation generation using GPT-3”, in *ASE*, ACM, 2022, 174:1–174:6. [Online]. Available: https://arxiv.org/abs/2209.02235.

[30] X. Zhou, K. Kim, B. Xu, D. Han, J. He, and D. Lo, “Generation-based code review automation: How far are we?”, in *ICPC*, IEEE, 2023, pp. 215–226. [Online]. Available: https://arxiv.org/abs/2303.07221.

[31] P. Ma, R. Ding, S. Wang, S. Han, and D. Zhang, *Demonstration of InsightPilot: An LLM-empowered automated data exploration system*, 2023. arXiv: 2304.00477 [cs.DB].

[32] Y. Xv, C. Peng, Z. Wei, F. Liao, and M. Xiao, “Can chat-gpt a substitute for urological resident physician in diagnosing diseases?: A preliminary conclusion from an exploratory investigation”, *World Journal of Urology*, vol. 41, pp. 2569–2571, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260247056.

[33] I. Mannstadt and B. Mehta, “Large language models and the future of rheumatology: Assessing impact and emerging opportunities.”, *Current opinion in rheumatology*, pp. 10–1097, 2023. [Online]. Available: https://journals.lww.com/co-rheumatology/abstract/9900/large_language_models_and_the_future_of.83.aspx.

[34] C.-H. Tsai, S. Kadire, T. Sreeramdas, *et al.*, “Generating personalized pregnancy nutrition recommendations with GPT-powered AI chatbot”, *Proceedings of the 20th International Conference on Information Systems for Crisis Response and Management*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261477927.

[35] X. Xu, B. Yao, Y. Dong, *et al.*, “Mental-LLM: Leveraging large language models for mental health prediction via online text data”, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260203216.

[36] Y. Cao and J. Zhai, “Bridging the gap – the impact of ChatGPT on financial research”, *Journal of Chinese Economic and Business Studies*, vol. 21, pp. 177–191, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258681864.

[37] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, *et al.*, *BloombergGPT: A large language model for finance*, 2023. arXiv: 2305.17564 [cs.CL].

[38] Z. Wang, Y. Li, J. Wu, J. Soon, and X. Zhang, *FinVis-GPT: A multimodal large language model for financial chart analysis*, 2023. arXiv: 2308.01430 [cs.CL].

[39] D. Z. Xiao-Yang Liu Guoxuan Wang, *FinGPT: Democratizing internet-scale data for financial large language models*, 2023. arXiv: 2307.10485 [cs.CL].

[40] Y. Yang, Y. Tang, and K. Y. Tam, *InvestLM: A large language model for investment using financial domain instruction tuning*, 2023. arXiv: 2309.13064 [q-fin.GN].

[41] A. Beheshti, J. Yang, Q. Sheng, *et al.*, “ProcessGPT: Transforming business process management with generative artificial intelligence”, *2023 IEEE International Conference on Web Services (ICWS)*, pp. 731–739, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259076468.

[42] M. S. Bernstein, G. Little, R. C. Miller, *et al.*, “Soylent: A word processor with a crowd inside”, *Commun. ACM*, vol. 58, no. 8, pp. 85–94, 2015. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2791285.

[43] K. Ferdowsi, J. Williams, I. Drosos, *et al.*, “ColDeco: An end user spreadsheet inspection tool for AI-generated code”, in *VL/HCC*, IEEE, 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/coldeco-an-end-user-spreadsheet-inspection-tool-for-ai-generated-code/.

[44] S. Gulwani, "Automating string processing in spreadsheets using input-output examples", in *POPL*, ACM, 2011, pp. 317–330. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/12/popl11-synthesis.pdf.

[45] X. Zhang, J. Li, P.-W. Chi, S. Chandrasegaran, and K.-L. Ma, "ConceptEVA: Concept-based interactive exploration and customization of document summaries", in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, Apr. 2023. DOI: 10.1145/3544548.3581260. [Online]. Available: https://doi.org/10.1145%2F3544548.3581260.

[46] K. Ferdowsi, R. Huang, M. B. James, N. Polikarpova, and S. Lerner, *Live exploration of AI-generated programs*, 2023. arXiv: 2306.09541 [cs.HC].

[47] S. E. Spatharioti, D. M. Rothschild, D. G. Goldstein, and J. M. Hofman, *Comparing traditional and LLM-based search for consumer choice: A randomized experiment*, 2023. arXiv: 2307.03744 [cs.HC].

[48] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel, *Nl2spec: Interactively translating unstructured natural language to temporal logics with large language models*, 2023. arXiv: 2303.04864 [cs.LO].

[49] I. Drosos, N. Wilson, A. D. Gordon, S. S. Ragavan, and J. Williams, "FxD: A functional debugger for dysfunctional spreadsheets", in *IEEE Symposium on Visual Languages and Human-Centric Computing*, Oct. 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/fxd-a-functional-debugger-for-dysfunctional-spreadsheets/.

[50] A. Reid, L. Church, S. Flur, S. de Haas, M. Johnson, and B. Laurie, *Towards making formal methods normal: Meeting developers where they are*, 2020. arXiv: 2010.16345 [cs.LO].

[51] D. Metaxa, J. S. Park, R. E. Robertson, *et al.*, "Auditing algorithms: Understanding algorithmic systems from the outside in", *Foundations and Trends® in Human–Computer Interaction*, vol. 14, no. 4, pp. 272–344, 2021, ISSN: 1551-3955. DOI: 10.1561/1100000083. [Online]. Available: http://dx.doi.org/10.1561/1100000083.

[52] C. Rastogi, M. Tulio Ribeiro, N. King, H. Nori, and S. Amershi, "Supporting human-AI collaboration in auditing LLMs with LLMs", in *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '23, Montréal, QC, Canada: Association for Computing Machinery, 2023, pp. 913–926, ISBN: 9798400702310. DOI: 10.1145/3600211.3604712. [Online]. Available: https://arxiv.org/abs/2304.09991.

[53] K. S. Jones and J. R. Galliers, *Evaluating Natural Language Processing Systems: An Analysis and Review*. Berlin, Heidelberg: Springer-Verlag, 1996, ISBN: 3540613099. [Online]. Available: https://dl.acm.org/doi/10.5555/547445.

[54] A. Salles, K. Evers, and M. Farisco, "Anthropomorphism in AI", *AJOB Neuroscience*, vol. 11, no. 2, pp. 88–95, 2020. DOI: https://doi.org/10.1080/21507740.2020.1740350.

[55] K. D. Evans, S. A. Robbins, and J. J. Bryson, "Do we collaborate with what we design?", *Topics in Cognitive Science*, 2023. [Online]. Available: https://doi.org/10.1111/tops.12682.

[56] G. Abercrombie, A. C. Curry, T. Dinkar, and Z. Talat, *Mirages: On anthropomorphism in dialogue systems*, 2023. arXiv: 2305.09800 [cs.CL].

[57] S. Amershi, D. S. Weld, M. Vorvoreanu, *et al.*, "Guidelines for human-AI interaction", in *CHI*, ACM, 2019, p. 3. [Online]. Available: https://www.microsoft.com/en-us/research/publication/guidelines-for-human-ai-interaction/.

[58] Q. V. Liao and J. W. Vaughan, *AI transparency in the age of LLMs: A human-centered research roadmap*, 2023. arXiv: 2306.01941 [cs.HC].

[59] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier", in *KDD*, ACM, 2016, pp. 1135–1144. [Online]. Available: https://arxiv.org/abs/1602.04938.

[60] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions", in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 1885–1894. [Online]. Available: https://proceedings.mlr.press/v70/koh17a.html.

[61] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions", in *NIPS*, 2017, pp. 4765–4774. [Online]. Available: https://arxiv.org/abs/1705.07874.

[62] C. Russell, "Efficient search for diverse coherent explanations", in *FAT*, ACM, 2019, pp. 20–28. [Online]. Available: https://arxiv.org/abs/1901.04909.

[63] B. Ustun, A. Spangher, and Y. Liu, "Actionable recourse in linear classification", in *FAT*, ACM, 2019, pp. 10–19. [Online]. Available: https://arxiv.org/abs/1809.06514.

[64] B. Y. Lim and A. K. Dey, "Assessing demand for intelligibility in context-aware applications", in *Proceedings of the 11th international conference on Ubiquitous computing*, 2009, pp. 195–204. [Online]. Available: http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/byl/publications/ubi205-lim.pdf.

[65] N. Tintarev and J. Masthoff, "Designing and evaluating explanations for recommender systems", in *Recommender systems handbook*, Springer, 2010, pp. 479–510. [Online]. Available: https://web.archive.org/web/20200710003624id_/https://blog.ag-nbi.de/wp-content/uploads/2015/08/08_TransparentRecommender.pdf.

[66] A. Sarkar, "Will Code Remain a Relevant User Interface for End-User Programming with Generative AI Models?", in *Proceedings of the 2023 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! '23)*, Cascais, Portugal: ACM, Oct. 2023, p. 15. DOI: 10.1145/3622758.3622882.

[67] J. A. Fails and D. R. Olsen Jr, "Interactive machine learning", in *Proceedings of the 8th international conference on Intelligent user interfaces*, 2003, pp. 39–45. [Online]. Available: https://3dvar.com/Fails2003Interactive.pdf.

[68] A. D. Gordon, C. Russo, M. Szymczak, *et al.*, "Probabilistic programs as spreadsheet queries", in *Programming Languages and Systems: 24th European Symposium on Programming, ESOP 2015*, Springer, 2015, pp. 1–25. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:810396/FULLTEXT01.pdf.

[69] A. Sarkar, M. Jamnik, A. F. Blackwell, and M. Spott, "Interactive visual machine learning in spreadsheets", in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, IEEE, 2015, pp. 159–163. [Online]. Available: https://advait.org/files/sarkar_2015_machine_learning_spreadsheets.pdf.

[70] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf, "Principles of explanatory debugging to personalize interactive machine learning", in *Proceedings of the 20th international conference on intelligent user interfaces*, 2015, pp. 126–137. [Online]. Available: https://openaccess.city.ac.uk/id/eprint/13819/1/.