Commons-Based Memories: Programming Practices and Large Language Models

Pierre Depaz

▶ **To cite this version:**

Pierre Depaz. Commons-Based Memories: Programming Practices and Large Language Models. Memory Studies Review, In press, 1 (2), pp.1-18. 10.1163/29498902-202400018 . hal-04796876

HAL Id: hal-04796876
https://hal.science/hal-04796876v1

Submitted on 30 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Commons-based memories: programming practices and large language models

Pierre Depaz

Dr., Université Sorbonne Nouvelle, Paris, France

pierre.depaz@sorbonne-nouvelle.fr

ORCID: 0009-0009-1489-247X

This article examines the role of digital interfaces in the creation and access of productive memory in the context of programming communities of practice. We first propose an analysis of question-and-answer platforms as the dominant means of creating such memories, and how such technical environment results in the constitution of knowledge commons. We then contrast these means with the emergence of programming-focused large language models, and how they represent a shift from large collaborative environment to non-human collaboration. As a result, we draw some implications regarding the sustainability of knowledge commons as well as the complementarity of digital devices in affecting the knowledge and values of the collective memories of programmers.

Keywords: programming, artificial intelligence, knowledge commons

Introduction

Programming has an ambivalent relationship to knowledge. Tightly linked to computer science, a formal discipline owing a large part of its origin to mathematics, programming is also the result of a set of *ad hoc* practices which have developed in a bottom-up fashion (Dijkstra, 1975). As such, the discipline has gone through a strict formalising process, called structured programming, but has also been the site of more informal knowledge exchanges; for instance, the development of software patterns in the mid-1990s involved the exchange of solutions to recurring programming problems, solutions which could be applied across projects (Gamma et al., 1995). These patterns, by definition motifs rather than laws, emerged from a knowing-how rather than a knowing-that, as the result of a collective effort to describe and disseminate shared ideas.

In parallel, programming knowledge has greatly contributed to, and benefitted from, digital communication systems. Strong advocates for a free circulation of information, understood as the basis of knowledge, programmers tend to share and pool these informal resources on various digital platforms, from Usenet boards to newsletters and wiki-style websites, resulting in the recurring motto "information wants to be free" (Levy, 2010). Programmers thus rely on shared networks of knowledge in order to accomplish their work, from remembering how to open a file in the Python programming language, to the commands which are to be typed in for the Git version control system, or asking peers what the best approach is to organise one's solution to a given problem. As programmers offload cognitive effort to their digital environment, this constant contributing to, and querying from, a shared database of information, ultimately constitutes the material existence of a socially- and technically-distributed cognition. We posit here that this computer-enabled distributed cognition, as memory existing out of the individual and into the machine, is one form of collective memory (Hollan et al., 2000).

Reticular writing, a writing understood as prominently networked, is the technical underpinning of a popular repository of collective programming memory. In the form of blogs, forums and question-and-answer websites, reticular writing enables information-pooling through online platforms (O'Reilly, 2005). This new form of collaboration led to the development of what

has been termed *knowledge commons*, understanding knowledge as a shared social-ecological system (Ostrom & Hess, 2006).

More recently, new developments in artificial intelligence research led to the release to the broader public of Large Language Models (LLMs), a specific kind of software operating on large swathes of online text, often anthropomorphised as *chatbots*, a kind of non-human actors involved in the dynamics of knowledge production (Latour, 1993). These LLMs offer an interface in natural language to query the knowlege contained in the corpus represented in the model, knowledge that is returned to the user as properly formed syntax. Since LLMs are trained on online corpora, they have benefitted as such from access to this knowledge commons. Furthermore, they have been presented as a solution to a programmer's issue of remembering technical facts and design solutions, by acting as their personal assistant.

It is this shift, from collaborative writing to programming assistant, that interests us. In this article, we intend to analyse the relationships between individual practices and collective memory, within the socio-technical context of crowd-sourced programming work. That is, we examine the conditions under which memory is created and accessed, paying particular attention to the mediations offered to the individual in order to access collective memory. Collective memory is considered here as a knowledge commons, constructed over time and transmitted across generations of individuals, and thus focusing more on the *knowledge* pole of collective memory, rather than on the *remembrance* one (Vinitzky-Seroussi, & Jalfim Maraschin, 2021). We focus particularly on the switch from one kind of *mnemotechnics* to another: from crowd-sourced, commons-based collective memory in the form of online forums and websites, to the use of LLMs. We thus hypothesise that, as different styles of archive, they also facilitate different communities of memory. How do LLMs change the way that collective memory of programming practice is being created and accessed? How do digital interfaces, as knowledge technologies, influence the relational existence of collective programming memory?

The approach we take in this contribution is at the crossroads of memory and science and technology studies, considering technology as one of the loci and repositories of human memory. Following Bernard Stiegler, we start from the *hypomnesis* of the human, its externalisation of memory through the inscription in a technological medium (Stiegler, 2010). This tertiary retention of human knowledge, in the form of discretised symbols stored in computer databases, presents

new challenges in the analysis of how memory is recorded, retained and accessed technologically. At the time of his writing, Stiegler presented this kind of retention as a form of reticular writing, which is also a networked reading (Bradley, 2021); our aim here is to highlight some of the effects of changing the medium of tertiary retention.

To answer these questions, we propose to take a comparative approach to the two socio-technical systems involved in the externalisation of programmer knowledge: the collaborative question-and-answer (Q&A) platform and the large language model. While our analysis of collaborative platforms will focus on the most popular ones – Stack Overflow and GitHub – and draw on previous work in the field of platform studies and netnography, the analysis of the LLMs, being more recent, will be based on a discourse analysis (Mullet, 2018) of both the producers of three major LLMs: OpenAI's Codex (and the related instantiations such as ChatGPT or GitHub Copilot), Amazon's CodeWhisperer, and Google's Bard. These were chosen as predominant actors in the field of programming assistants and will be examined in terms of the discourses that are deployed around these socio-technical objects, from the perspective of the corporate communication presenting such objects, and from the perspective of the users, and their commentary on the situated use of such interfaces.

We will start by establishing how, through the cases of the StackOverflow and GitHub discussions platforms, the collaborative model of collective memory reveals the intricacy of digital interfaces in the production and maintenance of such memories. We will then analyse how the three chosen LLMs position themselves in such a practice, highlighting how collective memory is assumed as the backdrop of individual performance and personalised assistance. Ultimately, we show how this shift in a socio-technical environment is, in the short-term, received, and how such a shift can lead to consequences in the lifecycle of digital-native collective memories as new non-human actors are introduced in communities of practice.

1. Collective memory in programming practices

Created in 2008 by two programmers, Stack Overflow is a question-and-answer website on which programmers ask questions, which can then be answered by members of the community. Other members of the community can then vote for the quality of answers, edit them, and comment on them. Stack Overflow is a particularly salient example of the surge in popularity in the software development community of these kinds of platforms (Anderson et al., 2012), a trend known as *social coding* (Storey et al., 2014). While most Q&A sites were initially aimed at providing useful answers to the individual asking the question, the massification of the use of these platforms has resulted in a shift towards question answering as a community-driven knowledge creation process whose end product can be of enduring value to a broad audience.[1]

Similarly, GitHub is a code repository platform, a service which was also created in 2008, and first aimed at providing an easy solution to the storage of source code. Along with this core technical feature, the platform offers some social features, such as user profiles, bookmarks and discussion pages. While GitHub also provided from the beginning a project-specific question-and-answer feature called *Issues*, which focuses on solving specific technical errors in a given project, they introduced *Discussions* in 2020, starting from the assumption that software communities' knowledge-exchange goes beyond the writing of source code, and also include brainstorming feature ideas, helping new users get their bearings, and collaborating on the best ways to use a software; essentially building a collaborative knowledge-base (Hata et al., 2021). Both Stack Overflow and GitHub are instances of what has been called the *Web 2.0*, where the technical infrastructure of the network allowed for the development of social networks in their contemporary form, ultimately acting as reservoirs of data for the training of machine learning models.

We chose Stack Overflow and GitHub as the two main platforms for our study due to their popularity. As of 2023, GitHub is the largest source code repository on the planet, with over 100 million users (Dohmke, 2023), while Stack Overflow boasts 20 million registered users and 100 million monthly unique visitors (Smith, 2022). On either platform, programmers ask questions about various issues they encounter, bugs, error messages, architectural decisions, programming languages, or more general inquiries (Abdalkareem et al., 2017). Besides their similarity in popularity and content, the platforms are also similar in their design: they allow registered users to

1 Stack Overflow has since evolved into the Stack Exchange network, providing community-orientated question-and-answers websites on topics ranging physical exercise to philosophy.

ask and answer questions, they allow non-verbal reactions to user contributions, and make all public questions searchable and accessible by non-registered users.

These websites operate as platforms, that is, as technical intermediaries where value is created by the putting-into-relation of users with one another. As such, they mediate and direct the possibility for interaction of their users, guiding their actions through design cues. Such design involves particular choices, slightly steering the user into desired behaviours in a form of *procedural rhetoric*, a term developed by Ian Bogost to suggest how the automatic enactment of rules expresses worldviews and achieve specific behaviours in the users (Bogost, 2008). In the case of Stack Overflow, such aim is to favour high-quality knowledge creation, and this takes place through the implementation of multiple designs, such as a voting system, user-generated tags, badges assigned to user profiles, and interfacing with popular search engines such as Google or DuckDuckGo (Barzilay et al., 2013). Similarly, GitHub deploys technical designs to support *social coding*, i.e., the realisation of productive programming work within a social media environment (Dabbish et al., 2012). For instance, the use of mentions, by which users can tag one another, allows contributors to call for the attention of specific programmers, deemed expert in the field, who could weigh in on the topic discussed (Zhang et al., 2016). In both cases, we see that the development of such collaborative knowledge-bases relies on the constitution of a socio-technical environment mediated by procedural interfaces.

The creation of value through these socio-technical environments has been referred to as commons-based peer-production by Yochai Benkler. This term describes a model of socio-economic production in which large numbers of people work cooperatively, often online, without clear or traditional hierarchy and seemingly without direct economic profit (Benkler & Nissenbaum, 2020). First, this peer production relies on the existence of a *knowledge commons*, as defined by Hess and Ostrom as a shared resource of information, to which all can have access, and which must be steered according to specific principles to maintain its quality (Ostrom & Hess, 2006). It also relies, as we have seen, on the infrastructure of platforms as mediators to organise the distributed and collaborative work. Through the socio-technical design of such platforms, Benkler and Nissenbaum have even argued that peer production encourages virtuous behaviours (Benkler & Nissenbaum, 2020), while Matei et. al. have argued that platforms such as Stack Overflow actually rely on the existence of the "sticky elite", a minority of expert programmers who

contribute disproportionately to the platform (Matei et al., 2017); digital designs do seem to influence slightly the cultural behaviour of members of the community.

However, contrary to more productive goods, such as open-source software, the value created on Stack Overflow and GitHub discussions emanates from non-rival, informational goods. Such goods are typical to the knowledge economy, in which the quality of the knowledge depends on its management by members of the community who have access and contribute to it.

Collective memory, as made up in part of information and knowledge of a given social group, is constituted here via a productive framework: such information and knowledge is related to the accomplishment of a task within a clearly defined context of programming communities of practice. Rather than long-term, shared collective memory, one which "travels from person to person through institutions, such as archives, and through communal mnemonic devices, such as monuments and street signs" (Margalit, 2004), question-and-answers platforms such as Stack Overflow and GitHub, a kind of *just-in-time* know-how as collective memory (Matei et al., 2017), is one whose circulation is accelerated, and whose sustainability is not guaranteed, by its digital media environment. Indeed, the process of having a question answered takes, on average, less than an hour (Bhat et al., 2014), while the access to this information is near-instantaneous, thanks to the platforms' tight integration with search engines through a process known as search engine optimisation.

Specifically, the kind of collective memory being deposited on those platforms is a form of social know-how, one which is prone to frequent change (through creation, duplication, edition and deletion of various levels of contribution). Contrary to an approach such as Wikipedia's, in which collective memory is created and maintained by the remembering of events throuh a similar technically-mediated social process (Grashöfer, 2014), the sort of collective memory we analyse here is focused on short-term practices (e.g., how does one solve a specific technical issue, how does one make an informed decision on the pros and cons of a specific way of doing things, how can one follow best practices of the programming communities). And yet, those short-term memories are also entangled with broader values and cultural exchanges.[2]

2 See, for instance, answers accepted for their cultural significance and lore rather than their strict technical accuracy.

We can therefore see the entanglement of this collective memory with both its technical access, through designed online platforms, and its intent, as short-term, technical retrieval enables the immediate accomplishment of a task. A particular example of the reliance of a programmer on this technical externalisation of productive knowledge can be seen in an April Fool's joke organised by Stack Overflow. The company suggested the commercialisation of a three-key keyboard, exclusively dedicated to copying and pasting Stack Overflow content into one's code base (Popper, 2021), parodying the widespread, folk joke that all knowledge needed by programmers is already in Stack Overflow and only requires mechanical duplication. And yet, Yang et. al. have shown that there are very little exact matches between source code excerpts posted on the Q&A platform and the public repositories of source code (Yang et al., 2017). So, while a Q&A website can act as a repository of common knowledge, interpretation and personalisation is still required for this information to once again become productive; this highlights online repositories not merely as collections of only fragments of information, but as a knowledge commons which then need to be modulated in specific instances.

This interface agency of Q&A platforms enabling retrieval of shared information has been upended by the rapid development of LLMs; both Stack Overflow and GitHub, as companies, have used their access to this trove of know-how in order to develop new interfaces to said know-how. In the case of Stack Overflow, they introduced OverflowAI Search, using LLM to retrieve knowledge as quickly and as accurately as possible (Bulajewski & Chan, 2023). GitHub's parent company, Microsoft, used their access to the largest source code repository on the planet, to develop Codex, a LLM which suggests source code answers to natural language queries (Zaremba, 2023).

In both cases, we are witnessing an alternative to a given model of collective memory formation through designed participatory web platforms, such as Stack Overflow and GitHub, to a model in which speed and productivity are used as standards justifying the development of a new kind of interface to these practical memories. In the next section, we develop on the nature and use of LLMs for memory retrieval, analysing the discourses around it to highlight how such interfaces are both presented to, and used by, programmers.

Large Language Models as memory interfaces

In order to approach the emergence of LLMs in the field of technical collective memory, the kind of memory most represented on Q&A websites, we focus on three specific models, considered the most efficient and the most high-profile. After explicating how such models work and insert themselves in programming practices, we then turn to a discourse analysis of how the models are being presented by the organisations who created them, and of how they are being received by programmers, thus highlighting not just a change in interface, but a shift in the values composing programmers' collective memory.

Large Language Models are a specific category of machine learning. They are focused on pattern recognition and generation after being trained on very large corpora, collected from online resources. They perform particularly well on text recognition and generation, such that they can act as natural language interfaces to the large datasets that they have been trained on (Zhao et al., 2023). From a technical perspective, LLMs involve several steps. First, they require the gathering of raw data to deduce patterns from. Then, such data is *cleaned*, meaning normalised to be comparable in a machine sense; in this process, some information can be lost from the original data.[3] Based on these normalised text inputs, a deep learning algorithm results in a language model, called *large* due to the sheer size of its training corpus. In the post-processing phase, the model is fine-tuned by humans, in which input/output pairs are being ranked in order to weigh desired behaviour more favourably than undesired behaviour. The resulting model can then be given a natural language input and provide a natural language output. For instance, given the input prompt "*What is collective memory?*", the LLM GPT-3.5 provides the output:

> *Collective memory refers to the shared pool of memories, experiences, and knowledge that a group of people, such as a community, society, or culture, hold in common. It is the way in which groups of individuals remember and recall events, stories, traditions, and cultural aspects that are significant to their identity and history. Collective memory can encompass a wide range of elements, including historical events, cultural practices, myths, symbols, and shared narratives.*[4]

3 Such as the design cues specific to Stack Overflow (number of upvotes or downvotes) or to GitHub (emoji reactions describing emotion towards a given text contribution).

4 Output generated on 21/07/2023, from https://chat.openai.com.

While they operate through a broadly stochastic method – generating the most likely sentence to be considered the answer to a query, their effectiveness in interacting under natural language terms[5] has led to several products being released to the public. We now turn our attention to three specific LLMs. These were tuned to help with programming tasks, by completing, improving and analysing source code. These three LLMs are: Codex, developed by OpenAI in conjunction with Microsoft; Bard, developed by Google; and CodeWhisperer, developed by Amazon. These three LLMs operate in a similar manner: providing a natural language input, they can generate source code, they can correct mistakes and optimise existing source code, and they can explain in natural language terms what a given excerpt of source code does (Bailey, 2023; Desai & Deo, 2022; Zaremba, 2023).

These three language models have been chosen by their production-ready nature (i.e., already in use by programmers through easy-to-use interfaces), as well as the fact that they were all trained on data including Stack Overflow questions-and-answers, thus tapping into this knowledge commons (Luu, 2023). While other LLMs exist to perform programming-related tasks, such as InCoder, CodeGen, PolyCoder (Liu et al., 2023), the aforementioned are the most visible in the programming community and thus are best suited to study their presentation to and reception from programming communities of practice.

Most of the communication and presentation around these models has been done through media announcements, either video or in the form of web pages. Here, we use Mullet's critical discourse analysis framework in order to highlight the implied values and intended uses in a set of discursive acts. We do so through the interpretation of various lexical references, tone of phrasing and intended audiences. We also pay attention to "*the work that language performs in society*" (Mullet, 2018), and specifically its role in framing the intended use and perception of a given part of this discourse and setting the values of a community of practice.

Our corpus is constituted by the institutional online presence contextualising the release of the three models, in the form of official websites (https://github.com/features/copilot, https://bard.google.com/ and https://aws.amazon.com/codewhisperer/), along with the blog posts and social media presence of their respective companies. Since each of these are technology

5 Hence their qualification as "*conversational agents*".

companies, we consider such an online presence to be the primary communication channel to their audience, and a canonical repository for the intents and values they aim to promote.

Each model follows the same mode of presentation and exist in the same historical context. Announcements regarding the release of the model is made via a blog post, detailing the features of the model via text, followed by the release of a standalone webpage on which a visitor can find similar features, but presented in a much more visually appealing layout. Since all of these models have been released within a year of each other (2022 for Codex, 2023 for Bard and CodeWhisperer), they exist in the same context of economic competition between the largest technology companies on the planet, willing to secure market shares resulting from technological innovation. We must therefore keep in mind that the discourses exist in both technical, marketing, and cultural realms.

First of all, we notice the themes of mysticism and collaboration in the names chosen of the LLMs. Codex and Bard respectively refer to European Middle-Ages manuscripts and Celtic storytellers and poets. A certain amount of designed uncertainty as to where these LLMs are getting their knowledge from is maintained. In the case of CodeWhisperer, the reference is made to the magical words which are to be uttered in order to achieve a complicated task, a reference to magic that can also be found in Google's naming of the next version of Bard as *Magi* (Grant, 2023). CodeWhisperer, Bard and Copilot (the user-facing name of the software relying on Codex) all suggest that these products are meant to assist and inspire the human in their writing task, presented as companions rather than replacements of the programmer using them. They are non-human technical apparatuses with whom the programmer can nonetheless be paired with.[6]

The second highlight lies in the values being promoted in the discourses. Across companies, the focus is on improving the efficiency of the programmer, as they present their code-focused LLMs as "a giant leap forward in developer productivity" (Amazon Web Services, 2023), "code faster, focus on business logic over boilerplate, and […] building great software" (GitHub, 2022), along with the more general "helping people with programming and software development tasks" (Bailey, 2023) by taking away the repetitive aspects of such tasks. The underlying ideas here

6 Such human-machine pairing through language interface has a long history, starting from the technical development of ELIZA, and cultural references that can be traced back to Ovid's Pygmalion (Jurafsky, & Martin, 2014).

are thus to get more things done faster and bypassing so-called unnecessary tasks. The speed at which the LLMs operate enable near-instantaneous knowledge-retrieval.

Finally, each of the LLMs advertise the tight integration in the current technical ecosystem within which programmers work. CodeWhisperer claims to be used in any code-related software, while Bard and Copilot are also integrated in the respective code-writing software of their parent companies: Google Collab and Microsoft VS Code. Recalling our previous discussion of the role of technical interfaces to influence the access to, or recall of, knowledge, we can see here a discursive hint at participating in a specific, exclusive technical ecosystem.[7]

Ultimately, since the LLMs highlight their presence "at the fingertips" (Amazon Web Services, 2023) of the programmers, they imply the bypassing of previous modes of remembering how to fix a bug, or how to write a menial operation; yet, because such memories of *know-how* were acquired through training on the original corpora, they actually obfuscate the fact that they are effectively replacing the previous contributors. This replacement, while conserving the concept of *just-in-time* knowledge production, nonetheless focuses on speed of access, rather than speed of creation, and thus setting aside the question of discursive context and of memory obsolescence. As LLMs emerge as new, alternative access to the knowledge commons created by programmers, the discourses deployed around LLMs by their creators reveal desires to position these new technical systems as personal assistants, improving productivity and seamlessly integrating into one's workflow, obfuscating the social relations that are at the origin of the creation of this collective memory. Beyond these claims are thus values and symbols that constitute an alternative ethos to StackOverflow and GitHub's social sharing.

Memories lifecycle and non-human collaboration

While the timeline of development and release of these LLMs limits the amount of data we have access to, and thus some of the predictions we can make as to the long-term impact of these tools,

7 Indeed, none of the LLMs are being presented as being interoperable with other LLMs, while Stack Overflow or GitHub draw value from their interoperability with search engines.

we can nonetheless sketch out a few early responses, as well as some dynamics and implications in the replacement of Q&A websites by LLMs in the process of retrieving programmer know-how.

Stack Overflow is explicitly mentioned by programmers in online discussions on the value of Copilot (Dembowsky, 2023; “Github Copilot – What’s Your Experience Been Like?” 2023; Hacker News, 2021). The main part of those conversations focuses on the benefit from taking away the menial tasks, therefore lining up with the discourses of the creators of the LLMs. One of the arguments made for using Copilot is that one would not want to lose time searching for an answer to a question online, nor want to copy and paste text from Stack Overflow into their program (Mutated_Zombie, 2022). However, since code found on Stack Overflow does not exist as a 1:1 copy in actual software, the argument here rather illustrates the possibility to substitute one technical system for another, drawing on playful cultural references from the collective memory of Stack Overflow users.

Another aspect of the discourse that emerges from these discussions is the process of acquiring knowledge, and the role that Stack Overflow plays in such a process. Several participants in these discussions have mentioned the different contexts at play: while LLMs focus on the individual’s context – meaning the source code currently written – Q&A websites focus on the answer’s context – meaning the different answers, comments, and design cues about the trustworthiness of the answers (Hacker News, 2021). The latter is considered to be important in order to understand the code at play rather than accessing more convenient answers, which also turns out too often to be erroneous (“Temporary Policy,” 2022; Vaithilingam et al., 2022).

Finally, a last significant aspect of the discussions between these two systems is that of the judgement of the community. Stack Overflow is notorious for being a community that is not friendly to newcomers, as individuals do not immediately know how to abide by the expected behaviours (Matei et al., 2017). As a result, some programmers report that they feel more at ease collaborating with a non-judgemental non-human rather than with a judgemental human counterpart (lezzgooooo, 2023). The advocated intimacy and personalisation of LLMs therefore seem to find some echo in how it is being used by programmers, as a companion or mentor in their practice of programming.

We have seen so far that the collective memory of programmers is made up, for a significant part, of technical know-how, and that such know-how is constituted into a knowledge commons through the mediation of specifically-designed interfaces. The replacement of such explicit interfaces with LLMs, consisting only of natural language, has been accompanied by a discourse composed of references to magic, productivity, companionship and seamless integration, as opposed to references to sharing, learning and social promotion in Q&A websites. As such, it is not only a change affecting technical means, but the communities of memory associated with it. With their discursive features, these technologies affect what is said, and to whom, and thus the collective encounters and representations of past knowledge. What we see here is therefore that the replacement of one means of accessing a collective memory also carries with it a set of values affecting this very collective memory. The use of one interface over another, which in turn gives access to a different representation of the same corpus, holds within it particular leanings in terms of power and social relations (Galloway, 2012).

The decline of traffic to the Stack Overflow website, losing 14% over a year, has been linked by some to the appearance of ChatGPT, an LLM sharing the same features and creator as Copilot (Carr, 2023). One of the explanations for this phenomenon could be the bypassing of new prompts for memory creation on Q&A websites; as newcomers and beginners rely on language models to solve simple tasks, this memory would effectively be transferred, in part, to the LLM, rather than externally to the Q&A website.[8]

A change of values is not the only collateral effect of a change in interface. As we have focused on the access of know-how on digital platforms, we must also pay attention to the creation of such know-how. In the case of Q&A websites, such know-how enters the knowledge commons through the process of asking a question, providing answers, and offering comments. In the case of LLMs, the process of contribution to collective know-how is less clearly defined. At first, LLMs are trained on corpora which, including Stack Overflow (Luu, 2023), take advantage of the knowledge commons. The process of updating such knowledge then relies either on further training on an updated corpus[9], or on simple binary feedback from the programmer. This eschews any further qualitative indication, such as distributed and aggregated votes, social reputation of the origin of the answer, or verbal contextualisation to be made accessible to others. In this case, these

8 Others suggest that such a decline in traffic is due to Stack Overflow's unwelcoming community (Notalabel_4566, 2023).

interfaces not only affect how memories are accessed, but also how they are constituted, and thus bring up questions of long-term sustainability. Through Q&A platforms, memory retrieval takes place contingently to memory creation, albeit sometimes by different individuals; through LLMs, memory retrieval and memory creation are disjointed, happening neither at the same place, nor at the same time. Interfaces can therefore be thought along these two lines of production and consumption, with different emphases put on either of these lines.

Conclusion

We focused here on the structural conditions of the lifecycle of collective memories, and how such inter-personal communication systems affect the recalling or the forgetting of such memories (Bastide, 1970). These structural conditions can have both a social component, in the values and expectations of behaviours present in the community, and a technical component, in the designed environments in memories are cretated, and retrieved; technical components can affect the organisation of social communities.

Programmers have created repositories of knowledge-focused collective memories online, technical memories which enable them to draw on past know-how to solve immediate problems. In order to create such a knowledge commons, the design of social interfaces plays an important role in steering agents towards the constitution of such commons, with both beneficial effects (e.g., sustained access to knowledge) and detrimental effects (e.g., intimidation of newcomers), through free mnemonic labour. The technical innovation of LLMs has provided an alternative to such collaborative memory models, bringing with it not just technical changes, but also particular values and assumptions. Paradoxically, such technical innovation relies on the existence of data (machine-understandable representations of knowledge) created by its predecessors, the social platforms. The reception by programmers of this new kind of interface has resulted in mixed feelings: beyond the correctness of answers, it is one’s stance on what constitutes meaningful or

9 Which can include the code written in the software to which the LLM is connected, and which the LLM might have helped write in the first place.

boring work, a judgemental interaction with a community or the bypassing of such judgement, which influences the perception of this new collaborative agent.

We can think beyond a simplistic replacement model which would see one means of accessing know-how (Q&A websites) by another (LLMs). Rather, we see the complex interplay of a digitally-mediated knowledge commons in which communities of programming practice have part of their collective memory being accessed by a new, non-human agent. From a designed environment which suggest behaviours for the agents that participate in it, we are now witnessing an alternative means of memory interface, focused on individual, rather than collective, interaction.

This shift from collective to individual interfaces does raise questions about the sustainability of this knowledge commons in its current state, mainly due to a lack of affordances to contribute to it in the current LLM interfaces. In the long-term, the bypassing of social interaction also hinders the archival process of new technical memories, asking questions about the sustainability of such interfaces. On the one hand, one could argue here for a technologically-facilitated forgetting of sorts, in which the private remains private. On the other hand, one could also see the emergence of LLMs as complementary non-human hybrids, changing the possibilities of interaction with collective memories rather than just narrowing them, or affecting the nature of the collective memories through biassed statistical sampling. Such change would, ultimately, be dependent on the value of communities of practice concerned along with the affordances of the technical systems they interact with, rather than strictly on the nature of its know-how.

References

Abdalkareem, R., et al. (2017). What Do Developers Use the Crowd For? A Study Using Stack Overflow. *IEEE Software*, 34(2), 53–60, https://doi.org/10.1109/MS.2017.31.

Amazon Web Services. (2023). AI Code Generator Amazon CodeWhisperer AWS. *Amazon Web Services, Inc.*, https://aws.amazon.com/codewhisperer/.

Anderson, A., et al. (2012). Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow. *Proceedings of the 18th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, 850–58, https://doi.org/10.1145/2339530.2339665.

Bailey, P. (2023). Code and Debug with Bard. *Google,* https://blog.google/technology/ai/code-with-bard/.

Barzilay, O. et al. (2013). Facilitating Crowd Sourced Software Engineering via Stack Overflow .In S. E. Sim & R. E. Gallardo-Valencia (Eds.), *Finding source code on the web for remix and reuse* (pp. 289–308)., Springer, https://doi.org/10.1007/978-1-4614-6596-6_15.

Bastide, Roger (1970). Mémoire collective et sociologie du bricolage. *L'Année Sociologique*, *21*, 65–108.

Benkler, Y., and Nissenbaum, H. (2020) Commons-Based Peer Production and Virtue (Reprint). *The Handbook of Peer Production*, John Wiley & Sons, 70–86, https://doi.org/10.1002/9781119537151.ch6.

Bhat, V. et al. (2014). Min(e)d Your Tags: Analysis of Question Response Time in StackOverflow. *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, 328–35, https://doi.org/10.1109/ASONAM.2014.6921605.

Bogost, I. (2008). The Rhetoric of Video Games. *The Ecology of Games: Connecting Youth, Games and Learning*, edited by Katie Salen, MIT Press.

Bradley, J. P. N. (2021). On the Gymnastics of Memory: Stiegler, Positive Pharmacology, and Illiteracy. *New Zealand Journal of Educational Studies*, 56(1), 5–22, https://doi.org/10.1007/s40841-021-00196-2.

Bulajewski, E., and Chan, E. (2023). *Behind the Scenes with OverflowAI Search - Stack Overflow*. https://stackoverflow.blog/2023/09/25/behind-the-scenes-with-overflowai-search/.

Carr, D. F. (2023). Stack Overflow Is ChatGPT Casualty: Traffic Down 14% in March. *Similarweb*, https://www.similarweb.com/blog/insights/ai-news/stack-overflow-chatgpt/.

Dabbish, L., et al. (2012). Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. *Proceedings of the ACM 2012 Conference on Computer Supported*

Cooperative Work, Association for Computing Machinery, 1277–86, https://doi.org/10.1145/2145204.2145396.

Dembowsky, C. (2023). Top Reasons NOT to Use GitHub Copilot · Community · Discussion #58562. *GitHub*, https://github.com/orgs/community/discussions/58562.

Desai, A., and Deo, A. (2022). *Introducing Amazon CodeWhisperer, the ML-powered Coding Companion | AWS Machine Learning Blog*. https://aws.amazon.com/blogs/machine-learning/introducing-amazon-codewhisperer-the-ml-powered-coding-companion/.

Dijkstra, E. W. (1975). *Craftsman or Scientist?* EWD40.

Dohmke, T. (2023). "100 Million Developers and Counting." *The GitHub Blog*.

Galloway, A. R. (2012). *The Interface Effect*. Polity.

Gamma, E., et al. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc.

GitHub. (2022). GitHub Copilot · Your AI Pair Programmer." *GitHub*, https://github.com/features/copilot.

Github Copilot - What's Your Experience Been Like? Worth It? *R/Webdev*, Mar. 2023.

Grant, N. (2023). Google Devising Radical Search Changes to Beat Back A.I. Rivals. *The New York Times*.

Grashöfer, K. (2014). Wikipedias Wissen. Vom Wandel Einer Mediengattung Als Bildungspolitischer Herausforderung. *Dichtung Digital. Journal Für Kunst Und Kultur Digitaler Medien*, edited by Roberto Simanowski, 43.

Hacker News. (2021). "The Amount of People Who Think That GitHub Copilot Is a Good Idea Is Just Fright... | Hacker News." *Hacker News*, https://news.ycombinator.com/item?id=27698153.

Hata, H., et al. (2021). GitHub Discussions: An Exploratory Study of Early Adoption. *Empirical Software Engineering*, 27(1), p. 3, https://doi.org/10.1007/s10664-021-10058-6.

Hollan, J., et al. (2000). Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research. *ACM Transactions on Computer-Human Interaction*, 7(2), 174–96, https://doi.org/10.1145/353485.353487.

Jurafsky, D., and Martin, J. H. (2014). *Speech and Language Processing*. Pearson Education.

Latour, B. (1993). *We Have Never Been Modern.* Harvard University Press.

Levy, S. *Hackers: Heroes of the Computer Revolution - 25th Anniversary Edition*. O'Reilly Media, Inc.

lezzgooooo. (2023). Stackoverflow Vs. ChatGPT Vs GitHub Copilot. *R/PinoyProgrammer*.

Liu, J., et al. (2023). *Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation*. arXiv:2305.01210, arXiv, https://doi.org/10.48550/arXiv.2305.01210.

Luu, M-L. (2023). Answer to "Was ChatGPT Trained on Stack Overflow Data?. *Artificial Intelligence Stack Exchange*.

Margalit, A. (2004). *The Ethics of Memory.* Harvard University Press.

Matei, S. A., et al. (2017). Do Sticky Elites Produce Online Knowledge of Higher Quality? *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, Association for Computing Machinery, 72–79, https://doi.org/10.1145/3110025.3110040.

Mullet, D. R. (2018). A General Critical Discourse Analysis Framework for Educational Research. *Journal of Advanced Academics*. 29(2), 116–42, https://doi.org/10.1177/1932202X18758260.

Mutated_Zombie. (2022). Thoughts on GitHub Co-Pilot? : R/Learnprogramming. *Reddit*, https://www.reddit.com/r/learnprogramming/comments/vjmdgu/thoughts_on_github_copilot/.

Notalabel_4566. (2023). ChatGPT Was Trained on Stackoverflow Data and Is Now Putting Stackoverflow Out of Business. : R/Webdev. *Reddit*, https://www.reddit.com/r/webdev/comments/15ai8ah/chatgpt_was_trained_on_stackoverflow_data_and_is/.

O'Reilly, T. (2005). What Is Web 2.0? *O'Reilly*, https://oreilly.com.

Ostrom, E., and Hess, C. (Eds.). (2006). *Understanding Knowledge as a Commons: From Theory to Practice*. MIT Press, 2006.

Popper, B. (2021). Introducing The Key - Stack Overflow. *Stack Overflow*, https://stackoverflow.blog/2021/04/01/the-key-copy-paste/.

Smith, C. (2022). Stack Overflow Statistics, User Count and Facts (2023) | By the Numbers. *Expanded Ramblings*, https://expandedramblings.com/index.php/stack-overflow-statistics-and-facts/.

Stiegler, B. Memory. *Critical Terms for Media Studies*, edited by W. J. T. Mitchell and Mark B. N. Hansen, University of Chicago Press, 2010.

Storey, M.-A., et al. (2014). The (R) Evolution of Social Media in Software Engineering. *Future of Software Engineering Proceedings*, Association for Computing Machinery, 100–16, https://doi.org/10.1145/2593882.2593887.

"Temporary Policy: Generative AI (e.g., ChatGPT) Is Banned - Meta Stack Overflow." *Meta Stack Overflow*, https://meta.stackoverflow.com/questions/421831/temporary-policy-generative-ai-e-g-chatgpt-is-banned, Nov. 2022.

Vaithilingam, P., et al. (2022). Expectation Vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, 1–7, https://doi.org/10.1145/3491101.3519665.

Vinitzky-Seroussi, V., & Jalfim Maraschin, M. (2021). Between remembrance and knowledge: The Spanish Flu, COVID-19, and the two poles of collective memory. Memory Studies, 14(6), 1475-1488. https://doi.org/10.1177/17506980211054357

Yang, D., et al. (2017). Stack Overflow in Github: Any Snippets There? *Proceedings of the 14th International Conference on Mining Software Repositories*, IEEE Press, 280–90, https://doi.org/10.1109/MSR.2017.13.

Zaremba, W. (2023). *OpenAI Codex*. https://openai.com/blog/openai-codex.

Zhang, Y., Wang, H., Yin, G., Wang, T., & Yu, Y. (2016). Social media in GitHub: The role of @-mention in assisting software development. *Science China Information Sciences*, 60(3), 032102. https://doi.org/10.1007/s11432-015-1024-6

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., … Wen, J.-R. (2023). A Survey of Large Language Models (arXiv:2303.18223). arXiv. https://doi.org/10.48550/arXiv.2303.18223