HAL open science

L'écriture guidée du code

Camille Roth, Jérémie Poiroux

To cite this version:

Camille Roth, Jérémie Poiroux. L'écriture guidée du code. RESET. Recherches en sciences sociales sur Internet, 2022, 11, 10.4000/reset.3429 . halshs-03904466

HAL Id: halshs-03904466

https://shs.hal.science/halshs-03904466v1

Submitted on 16 Dec 2022

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 - ShareAlike 4.0 International License

RESET
Recherches en sciences sociales sur Internet

11 | 2022
Codes. L'informatique comme elle s'écrit

L'écriture guidée du code

Le cas des algorithmes de recommandation

Guiding code development. The case of recommender systems

Camille Roth et Jérémie Poiroux

Édition électronique
URL : https://journals.openedition.org/reset/3429
DOI : 10.4000/reset.3429
ISSN : 2264-6221

Éditeur
Association Recherches en sciences sociales sur Internet

Référence électronique
Camille Roth et Jérémie Poiroux, « L'écriture guidée du code », *RESET* [En ligne], 11 | 2022, mis en ligne le 04 avril 2022, consulté le 23 avril 2022. URL : http://journals.openedition.org/reset/3429 ; DOI : https://doi.org/10.4000/reset.3429

Ce document a été généré automatiquement le 23 avril 2022.

© Association Recherches en sciences sociales sur Internet

L'écriture guidée du code

Le cas des algorithmes de recommandation

Guiding code development. The case of recommender systems

Camille Roth et Jérémie Poiroux

1. Introduction

1 La conception du code informatique est généralement appréhendée suivant deux grandes perspectives relativement complémentaires, selon que l'on s'intéresse aux conditions d'écriture du code (Brown, 2006 ; Mackenzie, 2006 ; Singer et al., 2010 ; Seaver, 2017) ou bien au lien entre ses aspects normatifs et descriptifs, c'est-à-dire entre procédures inscrites dans le code (Burrell, 2016 ; Ananny & Crawford, 2018) et leurs effets concrets et contextualisés (Cardon, 2015 ; Diakopoulos, 2015 ; Ziewitz, 2016). Malgré les appels récents à mettre en rapport les modes d'écriture des algorithmes et leur fonctionnement (Gillespie, 2014 ; Kitchin, 2017 ; Jaton, 2019), ces objets d'étude respectifs se recouvrent toutefois peu. Dans le premier cas, les travaux se focalisent sur les formes organisationnelles, notamment la configuration des activités d'écriture du code au sein des entreprises (Curtis et al., 1988 ; Button & Sharrock, 1996 ; Lethbridge et al., 2005) ou des communautés open-source (Kogut & Metiu, 2001 ; Mockus et al., 2002 ; Kelty, 2008). Dans le second cas, les objets d'étude sont plutôt liés à des types d'algorithmes, notamment de classement (Van Couvering, 2007), de prédiction (Vayre, 2018) ou plus généralement d'apprentissage automatique (Holstein et al., 2019), pour lesquels les effets régulatoires (Lessig, 1999), l'émergence de bulles de filtres (Pariser, 2011) ou de biais divers (Selbst & Barocas, 2015 ; Roth, 2019) font l'objet d'une attention particulière.

2 Nous ambitionnons ici de contribuer aux deux approches en nous penchant sur les algorithmes de recommandation et en articulant leur mode d'écriture à leur contexte d'utilisation. Ces algorithmes ont pour particularité d'être déployés sur des plateformes (Flichy, 2019) auprès d'un grand nombre d'utilisateurs, permettant ainsi une observation permanente de leur fonctionnement. Ils correspondent fréquemment à des milliers voire des millions d'utilisations : l'impact, en termes d'usage, des choix opérés

par leurs concepteurs est potentiellement du même ordre que celui des logiciels libres les plus répandus. La littérature sur les communautés de développeurs de logiciels libres met en lumière l'existence voire la nécessité de processus organisationnels éminemment bureaucratiques pour réaliser des programmes techniquement très complexes dans un cadre bénévole et auto-organisé (Cohendet et al., 2001 ; Mockus et al., 2002) : co-optation des nouveaux contributeurs, démocratie et méritocratie pour le choix des responsables de sous-projets, transparence des échanges, à l'instar du code, également ouvert.

3 Nous le verrons, ces caractéristiques contrastent sensiblement avec notre terrain, où les équipes chargées spécifiquement du développement de l'algorithme sont à la fois généralement de petite taille, aux compétences très variées et avec des attributions dépassant fréquemment l'écriture du code, qui n'obéit pas systématiquement à des processus formalisés. Le développement d'un algorithme de recommandation s'appuie d'une part sur des recettes considérées comme standards dans le champ et semble d'autre part laisser une place prépondérante à la surprise et l'imprévu. Les expérimentations et le tâtonnement se substituent souvent à l'implémentation de fonctionnalités définies à l'avance et selon un cahier des charges précis, tandis que le « bricolage » mais aussi l'introduction d'astuces sont monnaie courante. Les découvertes fortuites sont un aspect important du processus, via la lecture improviste d'un article scientifique ou une fouille de données exploratoire permettant de découvrir un phénomène surprenant. De fait, le développement des algorithmes de recommandation prend place dans un contexte de R&D proche, par certains aspects, de l'artisanat et de la recherche académique. Ce phénomène, en soi, n'est pas inattendu (Neff & Stark, 2004). Le cœur de notre contribution vise en revanche à montrer que les actions des utilisateurs servent de manière décisive, au sens propre, à guider l'évolution du code : elles sous-tendent ce que nous appelons différentes formes de *guidage de l'écriture*.

Le code homme-machine : usages et guidage

4 Nous allons notamment montrer que les conditions de production de ces algorithmes sont étroitement liées à leurs conditions d'utilisation : il s'agit d'un mode hybride où l'évolution du code dépend conjointement du travail des développeurs et de la quantification des actions des utilisateurs. L'argument suivant lequel ces actions peuvent être considérées comme une contribution productive à part entière est proche de ceux de Vayre (2017) et Casilli & Posada (2019), même si nous procédons d'une version encore plus affaiblie du sens du « travail » fourni implicitement par les utilisateurs. D'une part, classiquement, les traces d'usage contribuent aux données et paramètres dont se nourrissent les algorithmes, permettant ainsi des calculs non seulement au niveau individuel mais aussi au niveau de la plateforme toute entière, notamment via l'émergence de catégories et d'appariements agrégés à partir des actions de tous les utilisateurs (voir par exemple Davidson et al., 2010 ; Nguyen et al., 2014 ; Bonnin & Jannach, 2015 ; Beuscart et al., 2019). Cette approche est également au fondement de l'apprentissage machine itératif basé sur des données d'usage (Amershi et al., 2014). D'autre part, de manière beaucoup plus originale, les usages guident et valident de manière implicite les modifications introduites par les développeurs, dans le sens où sont retenues celles qui améliorent l'efficacité de l'algorithme mesurée auprès des utilisateurs. Bien que les codeurs expriment fréquemment et explicitement

des souhaits normatifs quant au type de comportement que leurs algorithmes devraient favoriser (notamment le fait de promouvoir une certaine sérendipité dans les contenus recommandés) nous n'en retrouvons pas la trace dans l'écriture du code au quotidien : à l'inverse, tout se passe comme si ces souhaits étaient absents des décisions concernant les évolutions du code. Les mesures d'efficacité reposent en outre sur des quantités classiques et principalement commerciales telles que l'audience, le taux de rétention, le taux de satisfaction, les clics, les revenus. Elles font peu débat parmi les développeurs et ne relèvent pas véritablement d'une démarche de quantification sociale (Desrosières, 1998) ou de benchmarking des agents ou organisations (Bruno & Didier, 2015), car s'appliquant aux performances agrégées de l'algorithme et étant ainsi de l'ordre du benchmarking du code. Nous n'étudierons toutefois pas ici la manière dont ces mesures d'efficacité sont discutées et sélectionnées, qui dépasse le cadre de cet article.

5 Par « écriture du code » nous entendons ici autant la conception du programme informatique et des processus clés de l'algorithme, au sens large, que l'écriture des lignes de code en tant que telles, sachant que ces deux catégories sont parfois poreuses. Ce terme recouvre donc essentiellement la manière dont les développeurs définissent et amendent l'algorithme afin de passer d'un cahier des charges général (par exemple, « recommander des contenus à des utilisateurs susceptibles de les intéresser ») à son implémentation (construire et combiner telle ou telle variable, introduire un calcul de tel ou tel type, décliner telle ou telle notion statistique ou concept d'apprentissage machine), dont l'introduction de variations (calculer telle grandeur de telle manière plutôt que telle autre, modifier le calcul en modulant la contribution de tel ou tel paramètre, etc.).

6 Nous mettons en évidence trois modes de guidage de l'évolution du code par les usages à des niveaux d'importance croissante.

7 Le premier mode s'apparente à un **« guidage de fonctionnement »**. Il s'agit d'intervenir ponctuellement en réponse à un comportement de l'algorithme jugé défaillant ou imparfait, notamment par la surveillance des usages et le *monitoring* du fonctionnement de l'algorithme qui en découle. Les développeurs sont alors susceptibles de bricoler à la volée le code qui peut « mal se comporter » (c'est-à-dire se comporter d'une manière qui diverge explicitement, au plus bas niveau, de ce qui a été prescrit : typiquement, corriger des bugs), voire d'intervenir dans le processus de calcul afin d'effectuer des actions hors de portée de l'algorithme (redémarrage des machines, ajout ou modification de bases de données internes ou externes).

8 Le second mode consiste en la production et plus précisément l'induction de catégories d'usage, afin d'introduire de nouvelles fonctionnalités ou variables. Il s'agit ici de ce que l'on pourrait appeler un **« guidage des catégories »** : le profilage des utilisateurs guide la découverte de nouvelles catégories sur lesquelles le code et l'algorithme sont susceptibles de s'appuyer. In fine, ce guidage contribue principalement à la *conception de nouvelles variantes* de l'algorithme et à la *modification des catégories* sur lesquelles il s'appuie.

9 Le troisième mode est le plus original et opère directement au niveau du guidage du développement : le succès d'une variante de l'algorithme est mesuré auprès des utilisateurs, en concurrence avec de nombreuses autres variantes et à l'aune des métriques d'efficacité mentionnées précédemment (liées aux objectifs commerciaux, à la satisfaction des utilisateurs ou à l'adéquation entre prédiction et adoption). In fine,

ce **« guidage des arbitrages »** concerne surtout *le choix entre les nouvelles variantes de l'algorithme*.

2. Propriétés de l'enquête

10 Notre protocole suit le déroulement standard d'une étude fondée sur des entretiens semi-directifs : définition du périmètre de recherche, recrutement des participants, entretiens, retranscription, analyse et présentation des résultats. Nous nous intéressons au développement des **algorithmes de recommandation**, dont le concept a été introduit par Resnick & Varian (1997), en nous focalisant sur ceux qui peuplent les plateformes : sites web ou applications qui sont généralement utilisés par un grand nombre d'individus pour interagir avec des contenus, des informations, ou d'autres individus. Nous avons adopté une définition large de ce type d'algorithmes, qui s'applique à tous les cas de l'étude : il s'agit d'un dispositif intégré à une plateforme qui peut être sollicité par les utilisateurs afin de leur proposer des biens et ou des services qu'ils n'ont pas cherchés en tant que tels : par exemple, obtenir des suggestions de morceaux de musique M_1, M_2, ... M_n semblables à un morceau M ou bien obtenir des recommandations d'hôtels H_1, H_2, ... H_n en toute généralité dans la ville V, contrairement au fait de chercher spécifiquement et explicitement le morceau M ou bien la page de l'hôtel H dans la ville V (ce qui correspondrait davantage à une démarche documentaire d'interrogation d'une base de données, appelée également *navigational query* par Broder, 2002). Les personnes avec lesquelles nous nous sommes entretenus sont toutes amenées, dans des proportions différentes, à contribuer à l'écriture du code d'un tel algorithme : elles se considèrent comme développeurs, concepteurs, designers, ingénieurs, « [*data scientists*] » et ont souvent un rôle de direction, sont responsables d'une équipe technique, « [*Chief Technical Officer*] » voire à la tête de l'organisation. Parmi les 188 personnes sollicitées, les femmes sont au nombre de 27 (15 %) mais seule une a accepté de participer à l'enquête ; l'échantillon final est donc quasi-exclusivement masculin. Par la suite, chaque répondant est identifié par un prénom anonymisé, le secteur auquel s'applique l'algorithme de recommandation auquel il contribue, et son âge (par bloc de cinq ans). En revanche, nous ne spécifions pas le métier qui est globalement le même du point de vue de notre étude : dans le reste de l'article, nous emploirons les termes de « concepteurs » et « ingénieurs » en tant que synonymes de « développeurs ».

11 Nous avons pu nous entretenir entre 2017 et 2018 avec 31 personnes appartenant à 29 organisations différentes œuvrant dans divers secteurs. Ceci nous permet d'éviter autant que possible de rendre compte de phénomènes singuliers potentiellement propres à une organisation précise. Les organisations se répartissent à parts relativement égales entre celles pour qui la recommandation est au cœur de leur activité économique principale (généralement plus jeunes et plus petites : des « start-up ») et celles pour qui il s'agit d'une fonctionnalité intéressante mais non-essentielle, sans que nous n'ayons observé de différences significatives dans les phénomènes relatés. À quelques exceptions près, nous n'avons pas recueilli différents discours provenant de la même organisation : deux entretiens ont eu lieu séparément avec deux personnes de la même organisation (Edwin et Gary), deux autres avec deux personnes ensemble et de la même organisation (Terence et Franck, puis Virgil et Sergio), et un entretien avec une personne au sujet de deux organisations (Jérôme).

12 Les entretiens ont porté sur les modes d'écriture et de développement du code, ses aspects organisationnels et les opinions des développeurs à leur égard, jamais véritablement sur ce qui pourrait relever du secret industriel et des procédures précisément inscrites dans tel ou tel algorithme. Le guide d'entretien n'a pas non plus porté spécifiquement sur les tensions pouvant émerger au sein de la collaboration. Lorsque les ingénieurs en ont volontairement évoqué, celles-ci ont principalement porté sur les choix stratégiques concernant l'introduction de telle ou telle fonctionnalité dans le produit, plutôt que les arbitrages dans l'implémentation de telle ou telle méthode ou l'exploration de telles ou telles données. Il s'agit certes d'une limite de notre terrain. Nous avons analysé les entretiens suivant les principes de la *grounded theory* (Glaser & Strauss, 1967 ; Lempert, 2007 ; Stol et al., 2016), par codage itératif sur trois boucles qui ont ainsi produit trois taxonomies successives de classement des discours. Parmi les catégories identifiées, trois sont particulièrement liées à l'hybridation entre analyse des traces d'usage et processus d'écriture du code ; elles sont ainsi au fondement de l'analyse qui suit.

Homogénéité des pratiques

13 La plupart des développeurs interrogés évoluent dans de petites équipes qui, malgré leur taille, présentent des caractéristiques à l'intersection entre communautés épistémiques et communautés de pratique (Cohendet et al., 2001). Ils évoluent en outre dans un champ professionnel et technique précis lié au développement des algorithmes de recommandation, qui partage un certain nombre de connaissances et de présupposés — en soi, une communauté épistémique ou de pratique dans un sens plus relâché, qui traverse et sous-tend toutes les organisations explorées ici. Nous n'avons cependant pas recueilli d'informations concernant leurs études ou niveau de diplôme. Les entretiens n'ont pas non plus porté sur les modes de socialisation ou de formation partagés par les développeurs : il nous est donc difficile de spéculer davantage sur l'origine des croyances ou références communes et sur l'appartenance possible à une véritable communauté épistémique.

14 Quoi qu'il en soit, nous avons observé une homogénéité marquée dans les discours décrivant les pratiques d'écriture du code, indépendamment de l'âge des répondants, du domaine d'activité ou de la taille de leur organisation. Par exemple, si nos entretiens ne permettent pas d'estimer en détail la proportion relative de chacun des trois modes de guidage évoqués précédemment dans le processus d'écriture, ceux-ci se retrouvent dans les discours de tous les ingénieurs, peu importe le produit ou le contenu recommandé (par exemple des nuitées ou des articles journalistiques), et peu importe l'expérience ou le secteur de l'enquêté.

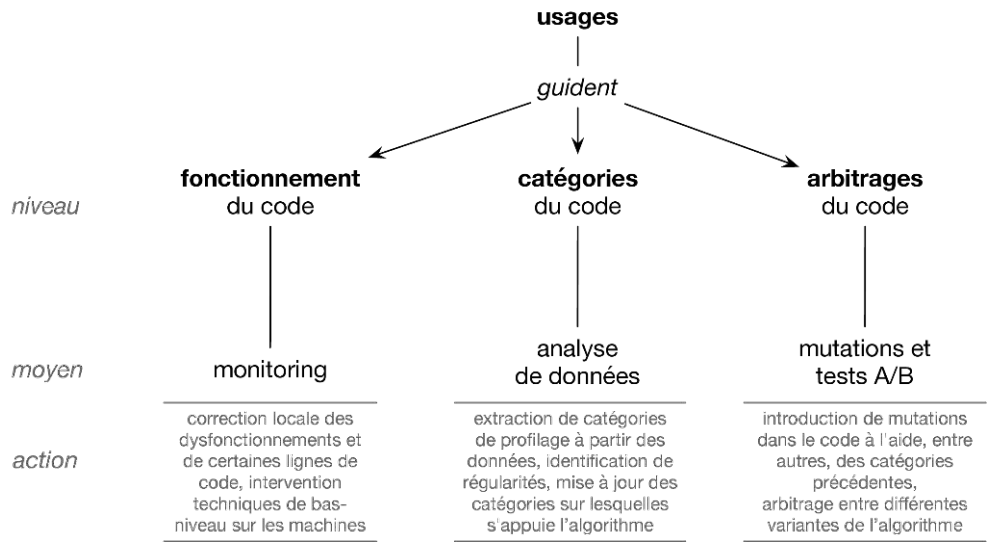
15 Cette homogénéité peut en outre s'avérer surprenante dans un domaine où la part d'innovation et donc d'originalité attendue est importante — les algorithmes, en tant qu'objet de recherche et de développement technologique, se devraient d'être des outils à la fois sophistiqués et originaux. Or, les enquêtés semblent s'accorder sur la simplicité et l'ubiquité des recettes qu'ils emploient pour réaliser le prototypage et la première mouture. Ces recettes combinent essentiellement des techniques éprouvées voire standards dans la communauté, au premier chef desquelles deux principales familles reviennent fréquemment : le filtrage collaboratif (Wang et al., 2015) qui utilise les signaux sociaux pour identifier des individus ayant consommé des contenus

similaires à celui de l'utilisateur et lui proposer parmi ceux-ci les contenus qu'il ne connaît pas ; et l'apprentissage machine basé sur la composition du contenu, dont notamment l'apprentissage profond (*deep learning,* voir LeCun et al., 2015) qui vise à élaborer des prédictions en entraînant des réseaux de neurones sur des données correctement « étiquetées » (i.e., en apprenant des associations passées pertinentes afin de prédire les associations futures). D'autres méthodes mentionnées sont moins fréquemment employées (dont l'inférence bayésienne, la régression logistique, ou encore le score de Wilson) mais elles semblent toujours faire partie d'un répertoire commun à tous et connu de tous (pour un tour d'horizon, voir Bobadilla et al., 2013) :

> « Les éléments de base, ils sont plutôt évidents et disponibles un peu partout. Le filtrage collaboratif fait ce que fait le filtrage collaboratif, si tu prends les taxonomies et tu compares essentiellement les éléments de contenu individuel selon leurs attributs, ce n'est pas sorcier. » (Jan, 50 ans, musique et vidéos) [1]
> « Un prototype qu'on a testé sur le petit jeu de données qu'on a comparé après à d'autres algorithmes que j'ai pris dans l'état de l'art des algorithmes de recommandations qui existent. » (Paul-Henri, 30 ans, musique)

16 Nous allons à présent montrer dans les trois sections qui suivent comment, après un prototypage initial, l'écriture du code se poursuit en lien étroit avec ses usages. Le schéma de la figure 1 récapitule les modes d'action des types de guidage identifiés, ainsi que les niveaux et moyens qui leur sont chacun associés.

Fig. 1 : Modes d'action des trois types de guidage de l'écriture du code par les usages.



3. *Monitoring* et guidage de fonctionnement

17 La contribution des usages au *fonctionnement* de l'algorithme se décline de deux manières principales. D'une part, afin d'assurer sa bonne exécution, lorsque les ingénieurs s'appuient en partie sur la surveillance régulière de l'activité de l'algorithme i.e., son « *monitoring* », afin de remplir un rôle d'assistance que certains enquêtés assimilent à celui d'un médecin ou d'un réparateur, rôle qui renvoie naturellement à la littérature foisonnante sur le travail de maintenance et de réparation des infrastructures (Denis et al., 2016). D'autre part, en tant que module algorithmique humain *in vivo* en remplacement d'une solution *in silico* inexistante ou trop coûteuse. Ce

rôle implique en outre parfois un cercle d'acteurs élargi allant au-delà des seuls concepteurs : développeurs et utilisateurs font alors office de rouage humain de l'algorithme (à rapprocher de la notion de *distributed agency* de Neyland & Möllers, 2017).

Diagnostics et remèdes : le « *monitoring* »

18 Dans le cadre de leur premier rôle, les ingénieurs ont effectivement une activité quasiment « médicale » en vue de maintenir le fonctionnement quasi-autonome de l'algorithme. Cela passe par des diagnostics ou par des remèdes, à l'instar d'un médecin qui suit un patient et prescrit des médicamentations ou des opérations chirurgicales. Le *monitoring* est ainsi une mission indispensable et généralement considérée comme évidente par les ingénieurs. Sur le papier, les algorithmes semblent souvent conçus de manière à être théoriquement complètement autonomes, ou du moins à pouvoir remplir leur tâche même en l'absence d'intervention, ainsi que le résume un codeur :

> « L'algorithme, il tourne tous les jours, tous les jours et il reprend les données du jour et il décide de nouvelles recommandations. Donc tu pourras le laisser tourner pendant des siècles et il n'y aurait pas de problèmes quoi, il marcherait toujours pareil donc il n'y a pas besoin de le faire évoluer. » (Baptiste, 35 ans, vidéos)

19 La plupart des répondants affirment néanmoins qu'une partie de leur travail consiste à s'assurer constamment que « tout se passe bien » — c'est-à-dire, pour le propos qui nous occupe dans cette section, qu'il n'y ait pas d'erreurs de fonctionnement pur, telles que des valeurs ou des temps de calcul aberrants liés strictement à une malfonction dans le code ou les données. Les usages contribuent ainsi en temps réel à ce que l'on pourrait apparenter à une forme de correction orthographique du code ou des données. En somme, il s'agit d'une surveillance, voire d'une ingérence bienveillante dans la routine algorithmique, exercée par le biais de métriques techniques et de tableaux de bord permettant de suivre usages qui en sont faits :

> « On a toujours des métriques diverses qui nous indiquent un peu la santé de l'algorithme et on peut avoir des petites alertes de temps en temps quoi et se dire 'tiens, il y a quelque chose qui ne marche pas bien là' ou on a des signaux bizarres donc on va le tester pour voir ce qu'il se passe ou alors l'arrêter parce que ça déconne complètement. » (Damien, 45 ans, musique)

20 Les ingénieurs développent souvent des algorithmes dont le fonctionnement peut être modifié manuellement à la volée, en jouant par exemple sur de simples coefficients. En « production », c'est-à-dire lorsque l'algorithme est déployé directement auprès des utilisateurs, l'essentiel du code est en effet certes fixe et figé, mais il est possible et prévu que les humains interviennent afin de le paramétrer pour l'adapter en fonction des besoins du moment. Selon les situations, les ingénieurs s'affairent alors à introduire des modifications, coder spontanément des remèdes ou améliorations s'ils rencontrent des problèmes de fonctionnement où s'ils sont insatisfaits des valeurs des indicateurs d'usage, même si l'algorithme est en mesure de fonctionner sans ces incrémentations.

> « Si tu essaies de faire quelque chose qui soit complètement automatisé, il y a encore trop de bruit, trop d'erreurs. On utilise l'automatisation pour changer d'échelle, mais il y aura toujours une supervision éditoriale ou des éléments éditorialisés pour améliorer les choses et s'assurer qu'on limite autant que possible les erreurs. » (Jan, 50 ans, musique et vidéos) *

21 Se dessine ainsi une alternance entre fonctionnement totalement autonome et séquences de maintenance déclenchées par des signaux directement liés à la collecte

continue de données d'usage. Une partie de la fréquence de ces ajustements manuels réguliers est liée au fait que les algorithmes de recommandation s'appuient sur des données qui évoluent constamment et qui exigent ainsi de ré-entraîner et d'ajuster périodiquement les algorithmes sur des données à jour.

> « La seule contrainte de cet algo c'est que ça sous-entend qu'on réindexe régulièrement les données (...) pour que l'index de recherche soit à jour. » (Edwin, 25 ans, objets culturels)

L'humain dans la boucle algorithmique

22 Le second mode d'intervention directe dans le fonctionnement est lié à l'implication des humains dans le processus computationnel lorsque l'algorithme est conçu explicitement pour faire effectuer une partie des calculs, qui sont généralement hors de sa portée, par des humains — une forme effective de cognition distribuée (Hutchins, 1995 ; Beunza & Stark, 2003). Dans ce cas, le périmètre opérationnel de l'algorithme n'est plus seulement délimité par le code et les bases de données qu'il consomme, mais il s'appuie en sus sur une collaboration humain-non-humain avec les ingénieurs ou d'autres acteurs : son bon fonctionnement repose sur des décisions humaines à des moments définis, comme si les acteurs humains jouaient le rôle de modules, « externes » à la machine mais intégrés dans le processus algorithmique. L'algorithme hybride alors la machine, qui exécute des lignes de code, et les humains, développeurs ou même non-développeurs, qui participent aux calculs, au sens large. Ces humains peuvent être les ingénieurs eux-mêmes mais aussi certains utilisateurs qui ont notamment des compétences éditoriales — par exemple, journalistiques ou musicales. On trouve par ailleurs une description très poussée de cette « symbiose » chez Morris (2015) et Bonini & Gandini (2019) dans le cas de la conception des listes d'écoute sur les plateformes de streaming musical, ou chez Hagar & Diakopoulos (2019) dans le choix des titres d'articles par les journalistes, où les employés humains collaborent activement avec les algorithmes.

> « C'est un algorithme qui est humainement piloté (...) [par] des personnes qui ont des compétences éditoriales (...) [pour] voir comment ce livre se catégorise, quels sont les livres qui pourraient y être rapprochés, enfin si vous voulez c'est un algorithme de cerveaux humains. » (Louis, 30 ans, livres)

4. Stéréotypes et guidage des catégories

23 Au-delà de ces actions à court terme, les usages contribuent en outre à définir et guider les *catégories* qui servent à concevoir ou amender les principes algorithmiques. Plus précisément, l'examen et l'agrégation des traces d'usage permettent d'induire des catégories de comportement, d'utilisateurs, de processus qui informent les développeurs au sujet des régularités (au sens de « *patterns* ») dans les interactions entre utilisateurs et plateforme.

Stéréotypes *ex ante*

24 Si la personnalisation occupe une place centrale dans la recommandation (Turow, 2012), les mêmes principes et routines, peu ou prou, s'appliquent à tous. Plus précisément, les paramètres et préférences individuels sont traités par le même

algorithme, même si les résultats de ces calculs sont évidemment susceptibles de varier d'une personne à l'autre. Sans grande différence avec les méthodes traditionnelles de mise sur le marché, les algorithmes de recommandation sont donc développés pour une cible assez large. À cet égard, les discours des ingénieurs renvoient généralement à une représentation *ex ante* plutôt homogène du comportement et des attentes typiques d'un utilisateur *moyen* à laquelle ils font fréquemment référence pour expliquer les grands traits des principes algorithmiques et de leurs objectifs. Cette représentation découle tout d'abord en partie de stéréotypes ou d'intuitions venant avec l'expérience du métier (Kitchin, 2014 ; Hallinan & Striphas, 2016). Plusieurs ingénieurs formulent notamment un certain nombre d'affirmations sans jamais évoquer de sources, mobilisant ce que l'on pourrait considérer comme des idées reçues (partagées au sein de l'entreprise ou de la profession) ou plus simplement des intuitions personnelles, qui fondent une connaissance parfois précise des attentes et souhaits des utilisateurs — une représentation proche du client rhétorique (« *rhetorical customer* ») de l'étude de Van Couvering (2007) sur la perception des utilisateurs par les concepteurs de moteurs de recherche. Certains évoquent ainsi des exemples typiques d'usage, souvent à la première personne, comme s'ils projetaient directement leurs propres manières de consommer et leurs attentes :

> « Enfin, comme si t'allais à la Fnac, et tu fais un peu ce que tu fais en allant voir la personne qui s'y connait, le disquaire. Bien, tu ferais pareil mais avec une machine quoi : "voilà, j'aime bien ce disque là mais il m'en faudrait un autre". » (Damien, 45 ans, musique)

Stéréotypes *ex data*

25 Certains ingénieurs reconnaissent toutefois qu'ils ne « savent pas qui sont les utilisateurs » (Edwin, 25 ans, objets culturels) ou qu'ils ne connaissent pas suffisamment leurs attentes, voire qu'un effort serait justement nécessaire pour en apprendre davantage :

> « Mais il y a aussi pas mal de travail qui se focalise beaucoup plus sur les utilisateurs. Donc : pourquoi les gens utilisent le site, quel comportement adoptent-ils lorsqu'ils interagissent avec le site, quel est leur but final, ce genre de choses. » (Gary, 45 ans, objets culturels) *

26 Classiquement, les traces d'usage représentent le carburant de la personnalisation où données, actions et paramètres individuels nourrissent des recommandations tout aussi individuelles. Ces signaux, disponibles et actualisés en permanence, sont soit implicites soit explicites (Zhao et al., 2018) : implicites lorsqu'il s'agit des traces de navigation et de comportement des utilisateurs (historique de navigation, temps passé sur certaines pages, achats, etc.), explicites lorsque les utilisateurs spécifient leurs préférences générales (par exemple, centres d'intérêt) ou donnent leur opinion sur certains contenus (notation, appréciation, désintérêt, etc.).

27 Par ailleurs, ces traces sont utilisées au sein de calculs effectués au niveau de la plateforme toute entière et permettant de faire émerger des catégories et des appariements agrégés à partir des actions de tous les utilisateurs (voir par exemple Davidson et al., 2010 ; Nguyen et al., 2014 ; Bonnin & Jannach, 2015 ; Hallinan & Striphas, 2016 ; Beuscart et al., 2019). Plusieurs ingénieurs affinent ainsi leurs croyances au sujet des utilisateurs en observant sans être vus, ce qui fournit un autre type

d'occasion de prendre conscience des problèmes et limites de l'algorithme actuel et d'en faire évoluer les catégories et principes sur lesquels il s'appuie :

> « On peut voir en fonction du comportement des auditeurs aussi, voir s'il y a un moment particulier où ils arrêtent d'écouter le service ou un moment particulier où ils nous donnent des signaux qui montrent qu'ils ne sont pas trop contents de ce qu'on leur propose. Et en fonction de ça, ça peut nous donner des idées d'algorithmes. » (Damien, 45 ans, musique)
> « L'algorithme tel qu'il était conçu dans la première version n'arrivait pas à adresser ce point-là (...). Donc on a une deuxième phase qui a été de clusteriser le profil utilisateur (...) On a réussi à déterminer que la cohérence était due aux artistes qui étaient écoutés ensemble, avec des graphes et de la clusterisation. (...) du coup on peut les regrouper ensemble .» (Quentin, 35 ans, musique)

28 De fait, les actions des utilisateurs alimentent aussi bien les recommandations en temps réel que les catégorisations projetées sur eux par les ingénieurs et susceptibles d'être intégrées plus tard dans le code. Les données d'usage sont ainsi exploitées quantitativement afin de constituer et de mettre à jour des « profils », c'est-à-dire des groupes d'utilisateurs aux comportements similaires, qui améliorent le fonctionnement de l'algorithme et les connaissances de leurs concepteurs à leur égard.

> « Les gens se rendent pas compte en fait qu'ils ont des comportements extrêmement moutonniers et que avec très très peu de profils utilisateurs on peut faire de bonnes recommandations avec quasiment tout le monde. Par exemple pour des films de cinéma, avec moins de 20 classes d'équivalence de personnes on fait des recommandations qui sont bonnes à 90 % .» (Jérôme, 40 ans, moteur de recherche et régie publicitaire)

29 De nombreux stéréotypes guident l'introduction de principes algorithmiques particuliers, qu'il s'agisse de caractéristiques de bas-niveau liées au matériel des utilisateurs, ou bien de la répétition de certains comportements d'une année sur l'autre :

> « Tu as de la reco qui peut se baser sur l'IP, sur ton type de navigateur, sur ton type de machine, et cetera genre les gens qui sont sur un Mac, ils ont plus de pognon (...) Tu retrouves les mêmes *patterns*. Genre tu sais qu'aux mois d'avril, mai tu peux faire remonter le Portugal en tête de liste pour les pages françaises. Ouais, c'est rigolo ? Mais c'est presque trop facile. » (Simon, 45 ans, hébergements)

30 Les interprétations et les stéréotypes « *ex data* » se mêlent parfois aux stéréotypes « *ex ante* » afin de compléter un modèle imaginé et spéculatif du comportement de l'utilisateur, et donc d'un algorithme pertinent. Le profilage basé sur les données s'entrelace alors avec les projections a priori afin de former un modèle hybride des utilisateurs, en sus des retours qualitatifs traditionnels, qui permet de préciser et de justifier la formulation des principes de guidage auxquels ils seront ensuite soumis. En ce sens, les développeurs semblent se placer comme des arbitres mieux à même de guider l'utilisateur que l'utilisateur lui-même et d'exprimer les critères normatifs qui permettront à l'algorithme de modifier ou « corriger » les attentes supposées imparfaites de l'utilisateur. Cette attitude n'opère pas nécessairement aux dépens de l'utilisateur, par exemple lorsqu'il s'agit de pallier leur myopie ou ignorance potentielle dans l'optique de « faire découvrir » :

> « Ce qu'on s'est dit, c'est qu'on allait demander les goûts des utilisateurs et qu'on allait lui recommander des livres qui étaient dans notre catalogue, qui étaient qualitatifs et qui étaient en adéquation avec ses goûts de lecture pour le pousser à lire, pour qu'il se rende compte que la lecture c'est pas juste du Musso et du Harry

> Potter mais qu'il y a d'autres auteurs extrêmement compétents qui peuvent écrire des choses qui lui plaîrait. » (Louis, 30 ans, livres)

31 Le discours de nos enquêtés à cet égard trahit néanmoins des représentations propres aux élites économiques et culturelles, pour lesquelles omnivorisme et éclectisme sont valorisés (Lahire, 2006 ; Coulangeon, 2011). Comme nous le verrons plus bas, ces critères normatifs certes socialement situés semblent toutefois n'avoir en pratique qu'une faible influence sur la direction que prend le développement du code.

5. Tests et guidage de l'arbitrage

32 L'introduction de nouveauté et de modifications dans le code reste naturellement, par construction, du ressort des développeurs, qui s'appuient ainsi en partie sur des idées et catégories suggérées par l'analyse des données d'usage. Les codeurs sont donc en charge de proposer des options d'évolution du code. À nouveau cependant, leur validation semble largement déléguée à l'analyse quantitative des usages observés, qui guident principalement les *arbitrages* entre les différentes options. Ce type de guidage de l'écriture est ici le phénomène le plus nouveau et relève une fois de plus d'une forme singulière de cognition distribuée entre codeurs, algorithmes et, certes à leur insu, utilisateurs. Elle peut être apparentée à un processus évolutionnaire où le rôle des développeurs semble parfois réduit à proposer des mutations dans le code qui seront retenues ou rejetées suivant la mesure de leur efficacité auprès des utilisateurs, où les techniques de validation des alternatives dites « A/B testing » (Kohavi & Longbotham, 2017) jouent un rôle essentiel.

33 On observe en effet principalement deux processus permettant de faire progresser le code. Le premier consiste à améliorer incrémentalement une solution particulière, soit à l'aide de l'analyse des données d'usage, comme indiqué précédemment, soit en mobilisant essentiellement des pratiques que les ingénieurs qualifient souvent de « bricolage », de « bon sens », de « bidouille », de « cuisine » ou bien encore d'« astuces » ou de « tweaking » :

> « Donc voilà, faut jouer avec ce qu'on prend. Une astuce comme prendre la racine carrée du résultat et puis ça fait un chiffre moins gros, typiquement. » (Maxime, 35 ans, informations)

34 En plus d'être médecins et portraitistes, les développeurs sont ainsi des cuisiniers qui expérimentent :

> « Si on est content tout va bien, et puis si on n'est pas content on bidouille pour que ça marche mieux la prochaine fois (...) On met des petits malus sur les anciens, on secoue tout ça. (...) On va ensuite saler tout le bazar en rajoutant de l'aléatoire puisque il y a que ça qui permet de s'assurer une certaine découvrabilité de choses qui sont en dehors du cercle des intérêts de l'utilisateur. » (Timothée, 25 ans, informations)

35 Le second mode consiste à mettre parallèlement en concurrence un certain nombre de variantes de l'algorithme employé jusque là et s'apparente à un balayage horizontal des approches possibles. En d'autres termes, on évalue plusieurs algorithmes ou plusieurs versions du même algorithme et on essaye simultanément différentes pistes de développement :

> « Et donc les métriques évaluatives fonctionnent sur un premier niveau qui est un niveau de masse, en fait, où (...) je balance des centaines de versions d'algorithmes, des centaines de combinaisons de paramètres. » (Pierre, 30 ans, sorties culturelles)

36 Bricolage et balayage sont deux modes essentiels que l'on retrouve dans le cas de la conception des algorithmes prédictifs (Vayre, 2018). Ils semblent plus largement liés au développement d'algorithmes opérant sur des données et des comportements seulement partiellement connus, où l'exploration des données joue aussi une place prépondérante, comme nous l'avons vu précédemment : nouveautés et améliorations trouvent largement leur source au cours de l'exploration des données d'usage et de découvertes relativement fortuites.

Le B.A.-BA de l'A/B testing

37 Ces modifications sont également validées de manière tout à fait singulière. Pour beaucoup d'enquêtés, cette validation s'appuie en grande partie sur les tests, qui font intégralement partie de la boucle de développement et semblent ainsi agir comme un membre de l'équipe voire un décideur. En effet, la décision de conserver une variante parmi toutes celles mises en concurrence simultanément est généralement déléguée aux résultats des tests sur les utilisateurs qui, de manière très originale, valident ainsi implicitement les modifications introduites par les développeurs. La quantification du comportement des utilisateurs et de leur réaction aux calculs et propositions des algorithmes de recommandation sert d'étalon à la validité des modifications introduites dans le code. Ce mode de guidage de l'écriture est aussi d'une toute autre nature que la notion d'apprentissage machine itératif (Amershi et al., 2014) où les modèles modifient leurs paramètres et s'adaptent au fur et à mesure des actions des utilisateurs.

38 La boucle évolutionnaire dont il est question ici implique le code de l'algorithme, et non ses paramètres. Nous l'avons vu, les développeurs sont chargés de produire plusieurs variantes plausiblement plus efficaces de l'algorithme en laissant une large part au tâtonnement et au hasard. Le test souverain va ensuite trancher parmi ces variantes, en se basant sur les comportements et les réactions observées auprès des utilisateurs. Le « A/B test » (Kohavi & Longbotham, 2017) constitue ici la pierre angulaire de ce mode de production dont le caractère hybride concerne potentiellement toute la chaîne de développement, du choix des hypothèses à tester en amont aux corrections à apporter en aval. Il consiste à confronter divers échantillons d'utilisateurs (généralement choisis aléatoirement) à plusieurs solutions (typiquement, « A vs. « B ») et de ne conserver que celle qui obtient le plus de succès. Ce succès doit généralement être statistiquement significatif, à l'aune de mesures ici aussi standards dans la communauté, comme les *valeurs-p* (qui traduisent le fait qu'une observation a une probabilité inférieure à *p* de se produire fortuitement). Nos enquêtés expliquent que la mise en place d'un test est « assez simple » et peut se faire rapidement : s'il est une procédure formelle dans l'organisation du travail de développement, c'est ce ping-pong routinier et permanent entre introduction de variété (les ingénieurs sont une source de nouveauté et de variations) et tri ex post (le test élague). Les développeurs emploient également des outils permettant de systématiser, dans une certaine mesure, l'évaluation des résultats des tests et des performances des variantes : la procédure est parfois tellement rodée que l'identification des meilleurs candidats peut elle-même être automatisée (pour une illustration technique et détaillée de cette démarche et de ces outils, voir également Xu et al., 2015) :

> « Alors en fait on a des plateformes de tests où on n'a plus qu'à plugger les algos. C'est des choses qui ont été automatisées. On va trouver les algos qui donnent les

> meilleurs résultats au prototypage et puis on va les transmettre après aux équipes de prod pour qu'ils réalisent le truc. » (Jérôme, 40 ans, moteur de recherche et régie publicitaire)

39 Traditionnellement, ces techniques ont été initialement proposées afin de mettre en compétition les résultats d'un algorithme donné (un processus qui est à rapprocher de la collaboration symbiotique évoquée plus haut, voir aussi Hagar & Diakopoulos, 2019) : par exemple, proposer deux versions de la page d'accueil d'un média en ligne mettant en exergue deux articles-phares distincts, à deux échantillons d'utilisateurs, puis, après une certaine période de temps, ne conserver que la version qui a rencontré le plus de succès.

40 Ici, c'est *l'algorithme lui-même* qui est mis en A/B testing :

> « Donc tout changement sur un canal touchant directement les clients devra faire l'objet d'un test A/B. Si tu ajoutes une nouvelle variable au système de recommandation, si tu ajoutes des recommandations à un endroit où il n'y avait pas encore de recommandation, (...) tout doit faire l'objet d'un test A/B. (...) On fait des expérimentations pendant une période de temps donnée, et on a toujours besoin de résultats significatifs avec p plus petit que 0,05 pour garder les résultats. » (Claes, 45 ans, restaurants) *

41 Les différentes variantes d'un algorithme peuvent être testées par les codeurs en interne et a priori (généralement au début du cycle de développement, sur des bases de données pré-existantes, en séparant arbitrairement des données dites d'entraînement de données dites « de test ») :

> « Ben le premier test, c'est le test de machine learning le plus classique du monde, on prend le passé ancien et le passé récent, on apprend sur le passé ancien, on tente de prédire le passé récent et on voit si on avait raison. » (Pierre, 30 ans, sorties culturelles)

42 ou bien à l'aide de *focus groups* composés de testeurs humains s'exprimant généralement au sujet d'un prototype relativement avancé (évaluation qualitative, critique, synthèse des idées d'amélioration) :

> « Et après si ça nous plaisait, on en parlait avec des gens, donc on avait un groupe de bêta testeurs qui était de 200 à 300 personnes et si ça leur plaisait on mettait ça en place. (...) On a fait des focus groups aussi on avait fait venir deux fois dix personnes (...) pour leur présenter des choses et pour leur demander ce qu'ils en pensaient. » (Thibault, 30 ans, lieux et sorties)

43 Toutefois, la plupart du temps, elles sont testées sur une portion des utilisateurs de la plateforme, en temps réel et à leur insu, souvent sélectionnés aléatoirement, et le gain de performance est mesuré quantitativement, par le biais des données d'usage, en comparaison d'un groupe témoin. Ce mode semble généralement correspondre au régime de croisière du développement de l'algorithme :

> « Avec le test A/B, on a en gros A : l'ancien algorithme (...) qui touche un certain ensemble d'utilisateurs ; et B : le nouvel algo qui touche un ensemble d'utilisateurs différents, plus petit. Et ensuite on compare principalement les résultats. » (Jan, 50 ans, musique et vidéos) *

44 Notons ici, pour nuancer notre propos précédent sur l'homogénéité des pratiques, que nous avons pu voir émerger un point de différenciation entre enquêtés concernant l'importance du déploiement des techniques dites d'« A/B testing », qui semble liée à la maturité du projet et à la taille de l'entreprise. Ces techniques sont en effet particulièrement pertinentes si elles sont alimentées par une certaine quantité de données de qualité, lorsque l'infrastructure entourant l'algorithme a également déjà

atteint un certain niveau de maturité. Les grandes organisations, avec beaucoup de moyens et parfois des équipes dédiées, l'utilisent ainsi très souvent pour tester le plus d'éléments possibles, les moyennes se limitent aux aspects centraux de l'algorithme, tels que des nouvelles fonctionnalités, tandis que les petites s'en servent de manière essentiellement parcimonieuse.

45 Les tests induisent ainsi un arbitrage empirique plutôt qu'argumentatif entre membres humains de l'équipe de développement. En plus du processus traditionnel de décision où l'équipe se réunit devant le tableau pour discuter les avantages et inconvénients d'une solution en argumentant sur ses caractéristiques techniques, le test semble être le décideur final qui « valide », à l'instar d'un manager, toutes les modifications et oblige les ingénieurs à abandonner les idées qui ne donnent pas de résultats suffisamment satisfaisants. L'évocation des « résultats » des tests renvoie à tout un travail de quantification de la part des développeurs afin de concevoir des outils de mesure permettant de définir ce qui constitue effectivement une amélioration. Ces métriques sont développées de manière ad hoc quoiqu'étant directement dérivées des critères d'efficacité commerciales évoqués précédemment.

46 Non seulement le test est inscrit comme une étape nécessaire du processus de développement, mais il semble même pousser les développeurs à s'en remettre intégralement à la sagesse des résultats afin de juger de la pertinence d'une nouvelle idée, plutôt que de tenter formellement d'en prédire l'effet avant sa mise en place. Nos enquêtés n'indiquent pas être gênés par cet arbitrage par les usages. Au contraire, ils semblent l'accueillir volontiers et sans hésitation :

> « Par exemple si on pense qu'un de nos robots apporte 5 % de vues en plus, alors dans la phase de prototypage, on va vérifier si c'est bien ce qui se produit avant de créer quoi que ce soit. Et si ça se produit on va construire le code. Mais si ça ne se produit pas, on va changer les paramètres, essayer d'autres trucs, ouais et si ça ne marche pas on va juste abandonner l'idée. » (Andres, 30 ans, informations) *

47 La chronologie décisionnelle paraît ainsi inversée : l'ingénieur ne semble pas faire particulièrement preuve de parcimonie a priori parmi les idées à mettre en œuvre, c'est le test qui triera a posteriori les meilleures d'entre elles, tandis que les utilisateurs jouent le rôle de cobayes :

> « On lance des algos sur la population et on voit après ce que ça fait (...) C'est important d'avoir de l'intuition pour la recherche, c'est bien mais il ne faut pas trop s'y fier. Il faut surtout tout tester (...) Il faut surtout envoyer beaucoup d'expériences parce qu'en pratique ça ne coûte rien et on peut faire des découvertes même des fois un peu fortuites, il faut être sûr de pouvoir faire le tour de la question. » (Ludovic, 40 ans, musique)

48 La collaboration homme-machine, cette écriture guidée que l'on pourrait qualifier d'augmentée par les retours implicites des utilisateurs, se joue ainsi à un niveau méta : tout se passe comme si les plateformes de test faisaient évoluer régulièrement et de manière presque autonome des variantes proposées par des ingénieurs humains qui, par le biais de mutations originales dans le code, semblent principalement chercher à *découvrir* des principes algorithmiques plus efficaces plutôt que de les *concevoir*. Il peut parfois sembler que les développeurs se contentent de formuler un grand nombre de variantes et laissent le soin au guidage de sélectionner celles qui sont les plus efficaces, se gardant de déconstruire véritablement l'assemblage complexe qu'ils construisent progressivement et d'ouvrir véritablement cette grande boîte noire, quand bien même cela serait effectivement possible. Ce processus est d'autant plus performant que les

utilisateurs fournissent en continu, implicitement et passivement, des informations concernant leur usage des algorithmes et, partant, leur satisfaction ou non à propos des résultats ou des recommandations qui leur sont fournis. Le discours des ingénieurs à cet égard est généralement justifié de manière bienveillante comme un moyen de proposer le meilleur service possible, afin que les utilisateurs consomment ce qui leur plaît — il s'agit de concevoir un algorithme en observant les utilisateurs pour précéder leurs souhaits et s'adapter à leurs goûts, ce qui peut entrer en contradiction avec le souhait affiché par ailleurs de favoriser la découverte et la sérendipité.

Grands et petits idéaux

49 Ce dernier point touche à deux autres aspects essentiels du processus de développement : d'une part, l'articulation entre profilage, stéréotypage et guidage des utilisateurs, que nous avons évoqué dans la section précédente, et d'autre part, le lien plus large avec les préoccupations normatives des développeurs — à savoir, vers quels idéaux devraient tendre les algorithmes qu'ils conçoivent. Dans le contexte de la recommandation, de nombreux enquêtés nous ont généralement fait part d'un idéal algorithmique qui favoriserait sérendipité et diversité (idéal qui, comme nous l'avons vu plus haut, n'est vraisemblablement pas sans lien avec un rapport à la culture socialement situé). Ces développeurs indiquent expressément souhaiter construire des mécanismes de recommandation qui amènent les utilisateurs à découvrir des contenus nouveaux et vertueusement diversifiés. Pour autant, absolument aucun d'entre eux n'en a mentionné de mesure effective — que ce soit statistiquement ou en évoquant des enquêtes auprès des utilisateurs.

50 Ces grands idéaux restent en pratique à l'état d'idéaux. Lorsque l'on ouvre la boîte de l'écriture du code, la connexion entre grands idéaux et les métriques simples d'optimisation mentionnées jusqu'ici, que nous pourrions appeler « petits idéaux », reste introuvable dans le travail quotidien. Pour décider des principes algorithmiques qui vont être finalement retenus dans le code, ne semble plus rester que l'auto-pilote du développement évolutionnaire guidé par les tests sous l'arbitrage de métriques simples. Certains développeurs conjecturent malgré tout que les petits idéaux peuvent occasionnellement ou fortuitement servir les grands, notamment si ceux-ci visent à remplir deux objectifs concrets qui pourraient être en apparence opposés, mais qui peuvent s'avérer finalement assez peu en tension : c'est-à-dire, surprendre l'utilisateur et lui donner ce qu'il cherche en vue de le garder sur la plateforme et d'avoir un service profitable :

> « Il faut bien comprendre qu'une entreprise qui fait du streaming se doit de faire découvrir de la musique aux gens, pas seulement pour une raison d'éthique etc. mais aussi parce que si les gens tournent sur leurs quatre albums favoris, ils n'ont aucune raison de payer dix euros par mois. Ils ont juste à acheter leur MP3 et basta. » (Ludovic, 40 ans, musique)

51 Nous n'avons néanmoins recueilli aucun propos qui fasse état de pratiques visant à vérifier cette conjecture.

6. Conclusion

52 La fabrique des algorithmes de recommandation obéit à un processus de développement guidé par les traces d'usage à trois niveaux, qui peuvent diversement se recouper en pratique : tout d'abord, assurer la bonne opération et les réparations quotidiennes sur l'algorithme, ensuite, découvrir les catégories qui vont guider l'introduction itérative de nouvelles variables voire de nouveaux principes algorithmiques, et enfin, sélectionner les principes algorithmiques qui répondent le mieux à des objectifs principalement commerciaux pour un type de plateforme donné. Simplement dit, l'utilisation du guidage algorithmique guide sa propre évolution. De fait, l'écriture de ce type de code apparaît à la fois comme un processus de conception et de découverte. On observe en effet que la plupart des ingénieurs conçoivent aussi bien des algorithmes qui implémentent des techniques standards de recommandation (telles que le filtrage collaboratif), qu'ils cherchent et découvrent des variantes optimales (au sens d'un minimum local ou global au sein d'un grand espace d'algorithmes possibles) par essais-erreurs. La rétention d'une version du code et donc d'une variation de l'algorithme obéit dans une large mesure à un processus évolutionnaire où la découverte est guidée par l'efficacité accrue apportée par telle ou telle mutation du code et constatée à l'aune des résultats des tests auprès des utilisateurs.

53 Ce constat est susceptible d'avoir des répercussions significatives sur la manière dont la politique des algorithmes peut être appréhendée. Prenons ici l'exemple canonique du classement du web incarné par le principe du « PageRank » : c'est-à-dire appliquer une notion de vote généralisé fondé sur les citations, ou liens entrants, entre les pages web. Si ce principe semble avoir contribué en grande partie au succès de Google Search, si les ingénieurs l'ont gardé, si les améliorations successives ont été conservées (et plus largement si ce principe joue encore en partie un rôle dans le calcul des résultats de recherche), ce n'est pas tant de droit, parce qu'il correspond à des choix de la part de Google, mais *de fait*, parce qu'il paraît avoir été plus efficace, auprès des utilisateurs de la fin des années 1990, que les principes de filtrage sémantique de concurrent comme Altavista. Il fallait donc *découvrir* plutôt que le classement des pages comme un vote généralisé. De ce point de vue, les concepteurs du PageRank n'auraient pas eu véritablement un immense pouvoir (notamment en imposant un principe de classement du web) mais « seulement » un peu de chance (notamment en trouvant un principe qui allait convenir à tant d'utilisateurs).

54 Plus largement, les principes algorithmiques introduits dans le code sembleraient inexorablement définis par le couple formé (1) par des critères d'efficacité commerciale standards (audience, ventes) et (2) par une certaine manière d'organiser la plateforme et la grammaire d'interaction des utilisateurs avec celle-ci (interface, design). En d'autres termes, la dynamique d'évolution du code serait directement le produit conjoint de l'architecture d'une plateforme et des variables à optimiser : à ce stade, ne resterait aux codeurs qu'un rôle essentiellement instrumental et limité, notamment si le point de départ du code s'appuie sur des approches largement partagées dans l'industrie. Dans l'optique d'une discussion de la politique des algorithmes, nous arguons ainsi qu'il ne serait finalement pas tant crucial d'ouvrir davantage la boîte noire de la conception du code en tant que tel, notamment par le biais d'études ethnographiques, au motif que peu de processus normatifs semblent se dérouler

véritablement à ce niveau-là. Plus précisément, il nous semblerait à présent au moins aussi important de proposer une ethnographie de la manière dont est défini et discuté le couple formulé ci-dessus, afin de comprendre (1) comment sont conçus les critères de performance et les outils utilisés pour guider le développement algorithmique, et (2) comment se décide le design des plateformes, dont notamment l'articulation des algorithmes sur celles-ci. De fait, la question de la politique des algorithmes apparaît principalement comme celle de la politique du design des plateformes, c'est-à-dire qu'il s'agirait à présent d'étudier les choix qui sous-tendent la mise en forme de tel ou tel mode d'interaction, autour de tel ou tel contenu, avec telle ou telle métrique d'optimisation, et donc avec tel ou tel critère de sélection des algorithmes.

Ce travail a été partiellement financé par le projet « Algodiv » (ANR-15-CE38-0001) financé par l'Agence Nationale de la Recherche et par le projet « Socsemics » (grant agreement number 772743) financé par le European Research Council.

BIBLIOGRAPHIE

Amershi, Saleema, Cakmak, Maya, Knox, William Bradley & Kulesza, Todd. *Power to the People : The Role of Humans in Interactive Machine Learning*, 2014.

Ananny, Mike & Crawford, Kate. « Seeing without knowing : Limitations of the transparency ideal and its application to algorithmic accountability », *New media & society*, 20 (3), pp. 973–989, 2018.

Beunza, Daniel & Stark, David. « The organization of responsiveness : innovation and recovery in the trading rooms of Lower Manhattan », *Socio-economic review*, 1 (2), pp. 135–164, 2003.

Beuscart, Jean-Samuel, Coavoux, Samuel & Maillard, Sisley. « Les algorithmes de recommandation musicale et l'autonomie de l'auditeur. Analyse des écoutes d'un panel d'utilisateurs de streaming », *Réseaux*, 213 (1), pp. 17–47, 2019.

Bobadilla, Jesús, Ortega, Fernando, Hernando, Antonio & Gutiérrez, Abraham. *Recommender systems survey. Knowledge-based systems*, 46, pp. 109–132, 2013.

Bonini, Tiziano & Gandini, Alessandro. « "First Week Is Editorial, Second Week Is Algorithmic" : Platform Gatekeepers and the Platformization of Music Curation », *Social Media + Society*, 5 (4), pp. 1–1, 2019.

Bonnin, Geoffray & Jannach, Dietmar. « Automated Generation of Music Playlists : Survey and Experiments », *ACM Computing Surveys (CSUR)*, 47 (2), p. 26, 2015.

Broder, Andrei. « A taxonomy of web search », *ACM SIGIR Forum*, 36 (2), pp. 3–10, 2002.

Brown, Barry. « 'The next line' : Understanding programmers' work », *TeamEthno-online*, 2, pp. 25–33, 2006.

Bruno, Isabelle & Didier, Emmanuel. *Benchmarking : L'État sous pression statistique*. Zones, 2015.

Burrell, Jenna. « How the machine 'thinks' : Understanding opacity in machine learning algorithms ». *Big Data & Society*, 3 (1), 2053951715622512, 2016.

Button, Graham & Sharrock, Wes. « Project work : the organisation of collaborative design and development in software engineering », *Computer Supported Cooperative Work (CSCW)*, 5 (4), pp. 369–386, 1996.

Cardon, Dominique. *A quoi rêvent les algorithmes : Nos vies à l'heure des big data*. Le Seuil, 2015.

Casilli, Antonio & Posada, Julian. « The Platformization of Labor and Society ». In Graham, Mark & Dutton, William H. (eds.), *Society and the Internet. How Networks of Information and Communication are Changing Our Lives*, Oxford University Press, pp. 293–306, 2019.

Cohendet, Patrick, Creplet, Frederic & Dupouet, Olivier. « Organisational innovation, communities of practice and epistemic communities : the case of Linux ». In *Economics with heterogeneous interacting agents*, Springer, pp. 303–326, 2001.

Coulangeon, Philippe. *Les métamorphoses de la distinction. Inégalités culturelles dans la France d'aujourd'hui*. Mondes vécus. Grasset, Paris, 2011.

Curtis, Bill, Krasner, Herb & Iscoe, Neil. « A field study of the software design process for large systems », *Communications of the ACM*, 31 (11), pp. 1268–1287, 1988.

Davidson, James, Liebald, Benjamin, Liu, Junning, Nandy, Palash, Van Vleet, Taylor, Gargi, Ullas, Gupta, Sujoy, He, Yu, Lambert, Mike, Livingston, Blake et al. « The YouTube video recommendation system ». In *Proceedings of the fourth ACM conference on Recommender systems. ACM*, 2010, pp. 293–296.

Denis, Jérôme, Mongili, Alessandro & Pontille, David. « Maintenance & repair in science and technology studies », *TECNOSCIENZA : Italian Journal of Science & Technology Studies*, 6 (2), pp. 5–16, 2016.

Desrosières, Alain. *The politics of large numbers : A history of statistical reasoning*. Harvard University Press, 1998.

Diakopoulos, Nicholas. « Algorithmic accountability : Journalistic investigation of computational power structures », *Digital Journalism*, 3 (3), pp. 398–415, 2015.

Flichy, Patrice. « Le travail sur plateforme — une activité ambivalente », *Réseaux*, 213(1), pp. 173–209, 2019.

Gillespie, Tarleton. « The relevance of algorithms ». In T. Gillespie, P. J. Boczkowski & K. A. Foot (Eds.), *Media technologies : Essays on communication, materiality, and society*. MIT Press, 2014.

Glaser, Barney G. & Strauss, Anselm L. *The Discovery of Grounded Theory : Strategies for Qualitative Research.* Aldine de Gruyter, 1967.

Hagar, Nick & Diakopoulos, Nicholas. « Optimizing Content with A/B Headline Testing : Changing Newsroom Practices », *Media and Communication*, 7 (1), pp. 117–127, 2019.

Hallinan, Blake & Striphas, Ted. « Recommended for you : The Netflix Prize and the production of algorithmic culture », *New media & society*, 18 (1), pp. 117–137, 2016.

Holstein, Kenneth, Wortman Vaughan, Jennifer, Daumé, Hal, Dudik, Miro & Wallach, Hanna. « Improving Fairness in Machine Learning Systems : What Do Industry Practitioners Need? » In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 2019, no. Paper 600 in CHI '19, pp. 1–16.

Hutchins, Edwin. *Cognition in the Wild*. MIT press, 1995.

Jaton, Florian. « "Pardonnez cette platitude" : de l'intérêt des ethnographies de laboratoire pour l'étude des processus algorithmiques », *Zilsel*, 1 (5), pp. 315–339, 2019.

Kelty, Christopher M. *Two Bits – The cultural significance of free software*. Duke University Press, 2008.

Kitchin, Rob. « Big Data, new epistemologies and paradigm shifts », *Big Data & Society*, 1 (1), pp. 1–12, 2014.

Kitchin, Rob. « Thinking critically about and researching algorithms », *Information, Communication & Society*, 20 (1), pp. 14–29, 2017.

Kogut, Bruce & Metiu, Anca. « Open-source software development and distributed innovation », *Oxford Review of Economic Policy*, 17 (2), pp. 248–264, 2001.

Kohavi, Ron & Longbotham, Roger. « Online Controlled Experiments and A/B Testing », *Encyclopedia of machine learning and data mining*, 7 (8), pp. 922–929, 2017.

Lahire, Bernard. *La culture des individus. Dissonances culturelles et distinction de soi.* La Découverte, 2006.

LeCun, Yann, Bengio, Yoshua & Hinton, Geoffrey. « Deep learning », *Nature*, 521 (7553), pp. 436–444, 2015.

Lempert, Lora Bex. « Asking questions of the data : Memo writing in the grounded theory tradition », In *The Sage handbook of grounded theory*, pp. 245–264, 2007.

Lessig, Lawrence. *Code : And other laws of cyberspace*. Basic Books, 1999.

Lethbridge, Timothy C, Sim, Susan Elliott & Singer, Janice. « Studying software engineers : Data collection techniques for software field studies », *Empirical software engineering*, 10 (3), pp. 311–341, 2005.

Mackenzie, Adrian. *Cutting code : Software and sociality*. Peter Lang, 2006.

Mockus, Audris, Fielding, Roy T. & Herbsleb, James D. « Two Case Studies of Open Source Software Development : Apache and Mozilla », *ACM Transactions on Software Engineering and Methodology*, 11 (3), pp. 309–346, 2002.

Morris, Jeremy Wade. « Curation by code : Infomediaries and the data mining of taste », European Journal of Cultural Studies, 18 (4-5), pp. 446–463, 2015.

Neff, G & Stark, D. « Permanently beta : responsive organization in the Internet era ». In Howard, PN & Jones, S (eds.) *Society Online : The Internet in Context*, SAGE Publications, pp. 173–188, 2004.

Neyland, Daniel & Möllers, Norma. « Algorithmic IF … THEN rules and the conditions and consequences of power », *Information, Communication & Society*, 20 (1), pp. 45–62, 2017.

Nguyen, Tien T, Hui, Pik-Mai, Harper, F Maxwell, Terveen, Loren & Konstan, Joseph A. « Exploring the filter bubble : the effect of using recommender systems on content diversity ». In *Proceedings of the 23rd international conference on World wide web. ACM*, 2014, pp. 677–686.

Pariser, Eli. *The filter bubble : What the Internet is hiding from you*. Penguin UK, 2011.

Resnick, Paul & Varian, Hal R. « Recommender systems », *Communications of the ACM*, 40 (3), pp. 56–58, 1997.

Roth, Camille. « Algorithmic Distortion of Informational Landscapes », *Intellectica*, 70 (1), pp. 97–118, 2019.

Seaver, Nick. « Algorithms as culture : Some tactics for the ethnography of algorithmic systems », *Big Data & Society*, 4 (2), 2053951717738104, 2017.

Selbst, A. & Barocas, S. « Big Data's Disparate Impact », *California Law Review,* 104, 2015.

Singer, Janice, Lethbridge, Timothy, Vinson, Norman & Anquetil, Nicolas. « An examination of software engineering work practices ». In *CASCON First Decade High Impact Papers*. IBM Corp., 2010, pp. 174–188.

Stol, Klaas-Jan, Ralph, Paul & Fitzgerald, Brian. « Grounded theory in software engineering research : a critical review and guidelines ». In *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 120–131.

Turow, Joseph. *The daily you : How the new advertising industry is defining your identity and your worth.* Yale University Press, 2012.

Van Couvering, Elizabeth. « Is relevance relevant? Market, science, and war : Discourses of search engine quality », *J. Comput. Mediat. Commun.*, 12 (3), pp. 866–887, 2007.

Vayre, Jean-Sébastien. « Une histoire de machines à recommander des biens de consommation : de l'Internet documentaire à l'Internet des données. Études de communication », *Langages, information, médiations*, 49, pp. 89–106, 2017.

Vayre, Jean-Sébastien. « Comment décrire les technologies d'apprentissage artificiel ? », *Réseaux*, 211 (5), pp. 69–104, 2018.

Wang, Hao, Wang, Naiyan & Yeung, Dit-Yan. « Collaborative deep learning for recommender systems ». In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1235–1244.

Xu, Ya, Chen, Nanyu, Fernandez, Addrian, Sinno, Omar & Bhasin, Anmol. « From infrastructure to culture : A/B testing challenges in large scale social networks ». In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 2227–2236.

Zhao, Qian, Harper, F Maxwell, Adomavicius, Gediminas & Konstan, Joseph A. « Explicit or implicit feedback? Engagement or satisfaction? A field experiment on machine-learning-based recommender systems ». In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing. Association for Computing Machinery*, 2018, SAC '18, pp. 1331–1340.

Ziewitz, Malte. « Governing algorithms : Myth, mess, and methods », *Science, Technology, & Human Values*, 41 (1), pp. 3–16, 2016.

NOTES

1. Les entretiens ont eu lieu en français ou en anglais : les extraits en anglais ont été traduits par nos soins et sont indiqués par une astérisque.

RÉSUMÉS

Plusieurs travaux récents sur les algorithmes de recommandation ont appelé à s'éloigner de l'étude de leurs effets, tels que l'émergence de biais de prédiction ou de bulles de filtres, pour se pencher sur la manière dont ils sont conçus. Nous proposons ici de répondre à cet appel grâce à une étude qualitative basée sur des entretiens avec une trentaine de développeurs. Nous montrons que les conditions de production de ces algorithmes sont très étroitement liées à leur utilisation. Déployés sur des plateformes auprès d'un grand nombre d'utilisateurs, permettant ainsi une observation permanente de leur fonctionnement, leur code évolue en effet d'une manière hybride qui dépend continuellement du travail des développeurs et des actions des utilisateurs. Simplement dit, l'utilisation du guidage algorithmique guide sa propre évolution – qu'il s'agisse d'introduire de nouvelles variables, de nouveaux processus algorithmiques et, surtout, de choisir entre de nombreuses variantes par le biais de tests quantifiant en temps réel les réactions des utilisateurs à l'aune d'objectifs essentiellement commerciaux. De ce point de vue, le développement du code obéit dans une large mesure à un processus évolutionnaire semi-autonome dont les tests auprès des utilisateurs sont le principal arbitre: les développeurs introduisent des mutations, les utilisateurs produisent implicitement le calcul de la performance, exprimés en termes commerciaux standards (audience, ventes). En soulignant l'importance cruciale du choix de ces métriques, une fois effectués les choix concernant l'architecture d'une plateforme donnée, nous appelons les futures recherches à formuler la question de la politique des algorithmes principalement sous l'angle de la définition de ces deux dimensions – performance et design des plateformes – plutôt que d'ouvrir davantage la boîte noire du code et de sa conception.

Several recent works on recommender algorithms have called for shifting the focus away from the study of their effects, such as the emergence of prediction biases or filter bubbles, to look at how they are designed. We propose here to answer this call thanks to a qualitative study based on interviews with about thirty developers. We show that the conditions of production of these algorithms are very closely linked to their use. Deployed on platforms with a large number of users, thus allowing a permanent observation of their functioning, algorithmic code evolves in a hybrid way that continuously depends on the work of developers and the actions of users. Simply put, the use of algorithmic thirty developers. We show that the conditions of production of these algorithms are very closely linked to their use. Deployed on platforms with a large number of users, thus allowing a permanent observation of their functioning, algorithmic code evolves in a hybrid way that continuously depends on the work of developers and the actions of users. Simply put, the use of algorithmic guidance guides its own evolution – whether it is introducing new variables, new algorithmic processes and, above all, choosing between numerous variants through tests that quantify user reactions in real time in the light of essentially commercial objectives. From this point of view, code development is to a large extent a semi-autonomous evolutionary process in which user testing is the main arbiter: developers introduce mutations, users implicitly produce performance calculations, expressed in standard business terms (audience, sales). By emphasizing the crucial importance of the choice of these metrics, once the choices concerning the architecture of a given platform are made, we call on future research to frame the question of algorithmic policy primarily in terms of the definition of these two dimensions –performance and platform design – rather than opening up further the black box of code and its design.

INDEX

Mots-clés : développeurs, algorithmes de recommandation, guidage, test A/B, catégorisation
Keywords : coders, recommendation algorithms, A/B testing, categorization

AUTEURS

CAMILLE ROTH

CNRS, Computational Social Science Team, Centre Marc Bloch, Friedrichstr 191, 10117 Berlin, Allemagne. roth@cmb.hu-berlin.de.
CAMS (Centre d'Analyse et de Mathématique Sociales, UMR 8557 CNRS/EHESS), 54 Bd Raspail, 75006 Paris, France.

JÉRÉMIE POIROUX

CNRS, Computational Social Science Team, Centre Marc Bloch, Friedrichstr 191, 10117 Berlin, Allemagne. poiroux@cmb.hu-berlin.de